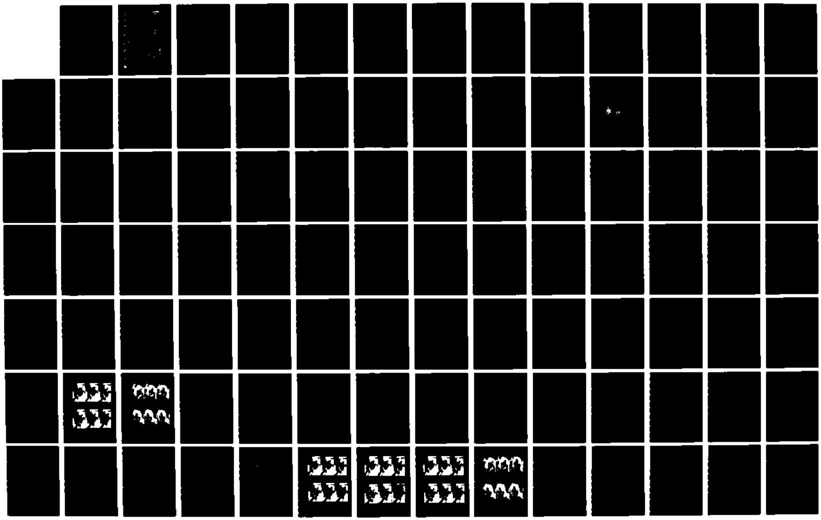
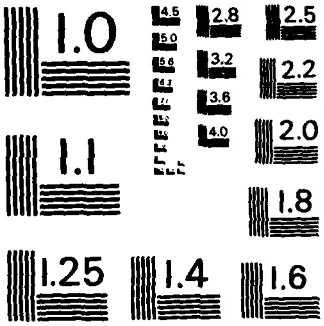
AD-A160 316 ADAPTIVE HYBRID PICTURE CODING VOLUME 1(U) ARKANSAS 1/2
UNIV FAYETTEVILLE DEPT OF ELECTRICAL ENGINEERING
R A JONES ET AL. 01 FEB 85 AFOSR-TR-85-0740
UNCLASSIFIED AFOSR-82-0351 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS – 1963 – A

UNIVERSITY OF ARKANSAS

ADAPTIVE HYBRID PICTURE CODING

Final Report, Volume I

by

Richard A. Jones
Principal Investigator

Carl D. Bowling
Senior Investigator

Department of Electrical Engineering
University of Arkansas
Fayetteville, Arkansas 72701

February 1, 1985

Prepared for the

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Under Grant No. AFOSR 82-0351

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. AFOSR-TR- 85-0740 ~~AFOS[illegible]0051~~	2. GOVT ACCESSION NO. AD-A160316	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Adaptive Hybrid Picture Coding		5. TYPE OF REPORT & PERIOD COVERED Final Report 1 Oct 1983 - 30 Sept 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard A. Jones Yogendra J. Tejwani Carl D. Bowling		8. CONTRACT OR GRANT NUMBER(s) ~~[illegible]~~ AFOSR-82-0351
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Electrical Engineering 232 SE Bldg. University of Arkansas, Fayetteville, AR		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2305/B3
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research Bolling Air Force Base Bldg. 410, Washington, DC 20332		12. REPORT DATE 1 Feb 1985
		13. NUMBER OF PAGES ~~345~~ 174
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release;
distribution unlimited.

17. DISTRIBUTION ST[illegible]ENT (of [illegible] abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY [illegible]TES

Different aspects of this report have been presented at Globecom, Miami, FL 1983, International Conference on Acoustics, Speech & Signal Processing, 1984, IEEE Trans on Systems, Man, Cyber., IEEE Trans. on Comm. Tech.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Compression, Quantizer, Shape Analysis, Feature Vector, Critical Points, Shape Space

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A system for the machine recognition of partial shapes is described. Shape analysis methods are reviewed in context to the problem of machine recognition of partial shapes, and their limitations.

The problem of defining the critical points for shapes and partial shapes with various degrees o[illegible] curvature is considered. It is shown that the critical points derived using criteria based on curvature alone are

FINAL REPORT, Vol. I. Grant No. 82-0351

ADAPTIVE HYBRID PICTURE CODING

Richard A. Jones
Principal Investigator

Carl D. Bowling
Senior Investigator

University of Arkansas
Department of Electrical Engineering
Fayetteville, Arkansas 72701

1 February 1985

Prepared for the Air Force Office of Scientific Research

MOTION COMPENSATED IMAGE CODING USING A PREDICTOR COEFFICIENT ENERGY CONCENTRATION MODEL

CONTENTS

LIST OF FIGURES

Figure Page

Chapter I
INTRODUCTION

With the introduction and proliferation of computers into all facets of the work place and the home environment, a new awareness of the capabilities and short-comings of the computer for various tasks has been found. The computer has proven very useful in performing repetitive, mundane tasks in offices and manufacturing process control environments, but lack of a good real-world/computer interface prohibits many uses. Presently a computers input connections to the real-world consist mainly of a keyboard and in some instances joysticks, graphics pads, light pens, and other sensors of the physical world. Recent research into this interface has provided the computer with 'ears', that is to say speech recognition. Not only can the computer hear, but it can also act upon human voice commands and speech. A 'voice' and associated language generation has also recently become a reality. The computer can generate syntactically correct language and then change this into intelligible human sounding speech. Perhaps the most important, and by far the most complex, interface would be the one which gives the computer 'eyes' or sight. Providing the computer with eyes and vision opens new realms for computer automation that in the past were either too difficult to perform blindly or completely impossible. This vision would allow the computer to perform difficult and tedious medical or industrial inspections at a much higher rate than is possible with human workers and also perform the inspections under hazardous or harmful environmental circumstances. With knowledge of the inspection, the computer could also make decisions, diagnoses and recommendations based upon its previous knowledge.

The possibilities for industrial assembly, combining robots with visual feedback, is staggering. With such devices, the jobs thought to be too boring and mundane or hazardous in the manufacturing field can be replaced by sighted robots that do not complain about the working conditions or monotony of the job. Just as computers and word processors replaced manual record keeping, filing and typing in the modern office to make it a rore enjoyable and productive working atmosphere, the seeing computer will do the same for the industrial and manufacturing fields.

A few of the many possible applications for intelligent seeing computers were given above, but by no means even scratched the surface as far as the uses that are currently envisioned or will be as the research evolves. As stated above, one of the earliest, and currently used, applications for this artificial vision is in the area of industrial and medical examination and inspection [1],[26],[52],[53],[107]. Some of the inspections are those which humans may deem overly tedious and boring, while other inspection methods performed by the computer can not be carried out with the speed and accuracy required, if even at all, by human workers. On a scale of difficulty, that of the visual inspection process would have to be very near the lower end of the scale. Very little, if any, interaction with the article being inspected takes place. Only a simple keep/return decision or perhaps a diagnosis or recommendation may be required. The problem amounts to providing the system with a 'good' product or example and then comparing the viewed products or specimen to be inspected against the good model. Provided all the tests and requirements are met, the piece will be passed or a good diagnosis returned. If any one of the possible visual tests fail, the article can then be rejected and possibly sent back for correction. In the medical case a diagnosis can be generated or recommendations for further tests can be given.

The system envisioned here is good for its particular task but lacks the intelligence to work outside the narrow area for which it was created. Perhaps more simply stated, the computer can not 'see' anything that it has not been programmed to see and hence its usefulness is extremely limited. The system can be improved with provisions to glean information from its field of view that represents physical properties of the objects being viewed and their relationship to the surrounding area. This gained intelligence will allow the computer to make judgements about its environment and alter it to meet specific goals.

This would allow the computer to perform simple manufacturing and assembly of multiple parts from various locations and orientations. This gain in ability comes at a high cost in increased complexity. This system is not programmed to see only one `object at one orientation, but several that must be recognized, tested and assembled into a product. Even this problem would have to be classed as a minor one when compared with the visual system that humans use in everyday life.

This marriage of image sensors to computers has already started, but the hurdles that must yet be crossed to reach what could be called 'intelligent machine vision' are both many and difficult. Both the computer and the sensor are here at present, but like a young child looking at the world he has never seen before, can not understand what he sees and hence cannot constructively affect the environment in which he lives. We have been able to give machines the eyes and brains to see, but have not yet been able to give them intelligence, understanding and vision. It is the aim of this work to provide the machine with a small bit of information, knowledge and understanding so that one day we may use what can truly be termed an intelligent seeing machine.

1.1 APPLICATIONS FOR ARTIFICIAL VISION

The possibilities and end uses for a computer that can intelligently see are endless. There has been much written in the general area of machine intelligence and more specifically machine vision. A scan of the references will provide many articles on various topics dealing with artificial intelligence and vision. At present the systems that actually exist and are in use are few and far between, but their number is growing and uses expanding. As a general rule, sighted computers can be divided into two major categories. The first category consists of systems that simply inspect their respective input images for flaws or abnormalities [1],[51],[53],[86], [107], and [145] and make decisions based upon these investigations. The second category extends the first in that it adds feedback to change its environment and hence act upon what it has seen [61],[67],[77] in such a way that it uses vision to constructively interact with its environment.

Many scientific disciplines have sought solutions for questions associated with vision. What is vision? How is vision defined? How is vision interpreted? These are just a few of the endless number of questions that arise when dealing with the topic of vision. Psychologists look to the internal thought processes to formulate theories about human interpretation of vision through visual stimulation. Physiologists on the other hand, are concerned with what biological processes are necessary for the generation of the synaptic signals associated with the sensations of vision.

From research carried out by the disciplines above, it has been found that the human visual system is a very complex and complicated network of biological subunits. Some examples of these subunits are the light receptors, i.e. the rods and cones, the optic nerve for visual transmission, and the brain with associated memory for interpretation. Each of these subunits constitute a very complex system in and of itself. So it is not surprising to find that when the biological visual system is *modeled* by *hardware* devices and *software* that the nonbiological visual system will also be a very complex and complicated set of subunits.

Just as the human visual system divides the whole of vision into many separate but related parts, so then should an artificial vision system. The human eye is not simply a sensor that transforms light into electrical signals, but also an enormous parallel processor that performs many of the lower level visual discrimination functions. Many of the rods and cones of the retina are specially sensitized so as to fire only for very specific physical occurrences. Examples would be those that fire only when an object moves at a specific speed in a specific direction, those that fire only when an object accelerates and many others related to various kinds of motion. There are also those which fire only when stimulated by an input field containing vertical or horizontal lines or boundaries. With this it can be seen that the eye does a major amount of processing before the signal even enters the optic nerve. The same cannot be said about the camera used for a computer's eye. By the time the visual signal reaches the brain a very large amount of image processing has already taken place. The brain receives the information in a form far different from that which the eye detected. The information content may be nearly the same, but the actual amount of data is far different. The sensors, optic nerve and other elements that connect the eye to the brain have performed what is termed data compression.

1.2 IMPORTANCE OF IMAGE CODING

Of mans five senses, vision would have to be termed the most data intensive. We, as sighted humans, tend to take the amount of information processing done by our visual system for granted. As was stated in the previous section, the amount of data that reaches the brain is far smaller than the amount of data that is actually incident upon the retina. This data reduction or compression that takes place prevents a computational overload on the brain by delegating some of the lower level visual functions to other parts of the anatomy. In a similar manner, artificial vision needs a means by which the amount of data can be reduced. It is at this point that the similarity in the structure of the human and artificial visual systems must stop. The electronic camera is a much simpler device than is the human eye, and hence processing that occurs in the eye must, in the machine model, be centrally processed. For this reason, and perhaps the alleviation of computational complexity, an intelligent visual system will also require a means for data compression.

An intelligent vision system is not the only application requiring the need for data reduction of imagery data and computational requirements are not the only resources strained by the very large amount of data present in imagery. Storage, transmission and system complexity are all adversely affected by the high data rates present. Many ideas remain on the drawing board or in limited use because of the expense that this large data rate entails. The ideas of digital television, video phones and facsimile have been with us for sometime, but the enormous bandwidths associated with each make them impractical when it is realized that bandwidth, like many material resources, is limited. For these and other similar products it can be seen that if commercial viability is to be obtained, then the video bandwidth requirements must be drastically reduced and this is achieved through data compression.

Applications of data compression are not limited to the commercial market alone. The military and space scientists also have many applications that for one reason or another may require data compression. On earth if new bandwidth is required it is a simple task to run another cable or optical link, but for space it is much more procedural plots, each pair consisting of two different gain values. Figures 2.4, 2.6, 2.8, and 2.10 consist of methods which require forward and inverse transformations at each iteration while the remaining plots depict a method requiring retransformation only on block intervals. Figures 2.4 through 2.7 involve the pel recursive displacement estimation approach while 2.8 through 2.11 involve coefficient recursive displacement estimation. All of the plots are similarly constructed where the horizontal, or X axis, represents the iteration number or the number of times equation (2.12) was executed. The vertical, or Y axis, represents the displacement error in pixels. The test images used were displaced by 2 pixels and the initial guess for the displacement was assumed to be zero, hence the initial displacement error of 2.

The data compression that is achieved by the system can be attributed to two major characteristics of the system and the data. The first is due to the redundancy removal that is obtained by the prediction process and the second is due to the fact that many of the transform coefficients can be grossly quantized or completely neglected with acceptable results.

One of the major problems of coefficient recursive estimation is that it is very scene and position dependent. In one scene or position the displacement may converge in as few as 4 or 5 iterations where for others it may fail to converge at all. Another problem may prove to be the choice of a good value for the gain factor, epsilon. As figures 2.4 through 2.11 verify, the choice of the gain factor plays an important role in the proper convergence of the algorithms and does not seem to be related to the image itself. The FORTRAN computer program used to test the algorithm and generate the iteration plots is contained in appendix A.

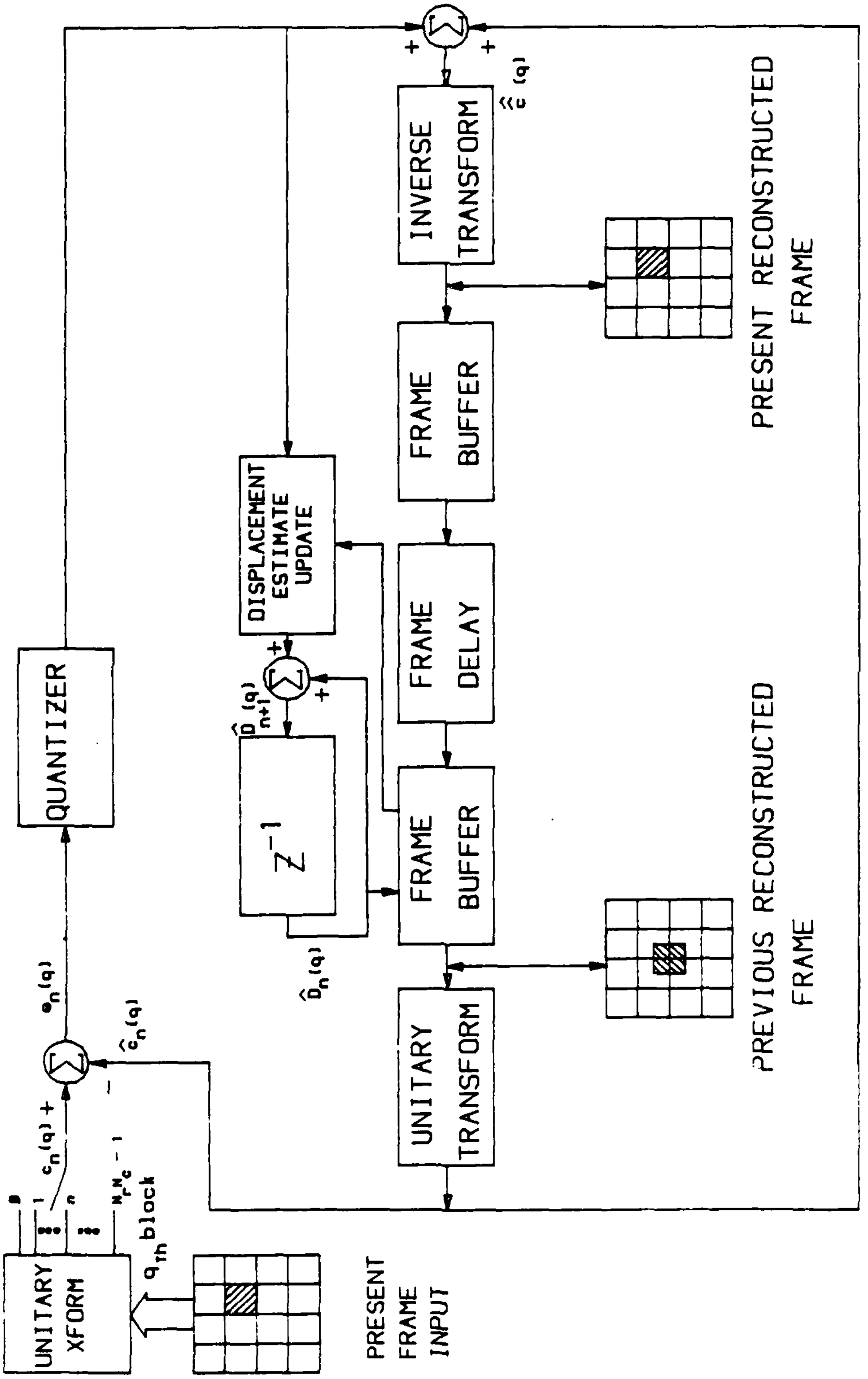


Figure 2.3 Motion Compensated Hybrid Transform-DPCM coder-decoder Pair.

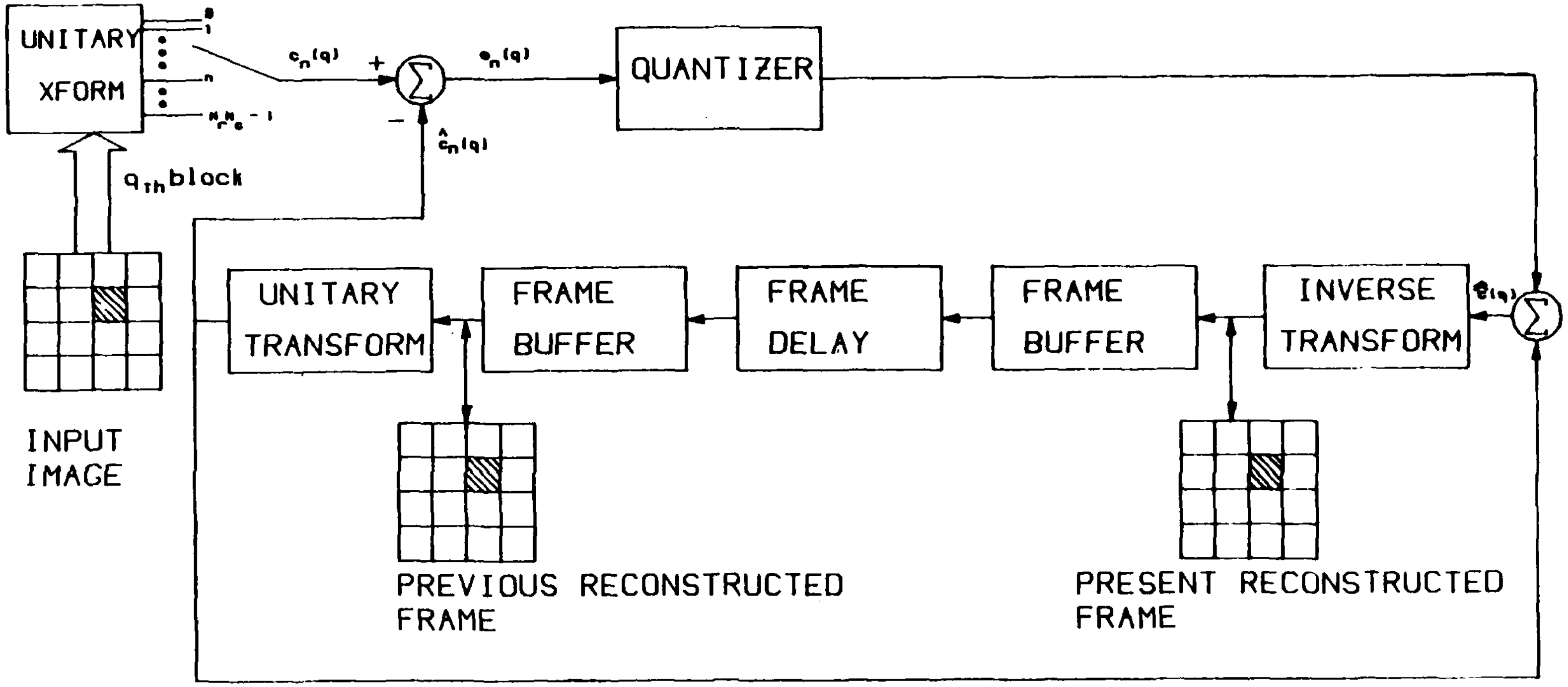


Figure 2.2 Hybrid Transform-DPCM coder-decoder Pair.

Results of this and the earlier pel-recursive method, in term of the displacement error, are given below for various values of the iteration gain. The test image is a radially decaying cosine function of radius 60 and peak-to-peak amplitude of 220 at the center decreasing to 130 at the circumference. The period also decreases radially, starting with a period of 20 pixels at the center and ending with 10 pixels at the edge. The equation used to generate the test image is

$$I(R) = 100\exp(-0.01R)\cos(2\pi R/P) + 128. \tag{2.14}$$

The displaced image is shifted two pixels in the x direction between time frames. Figure 2.1 is a picture of the actual test image.

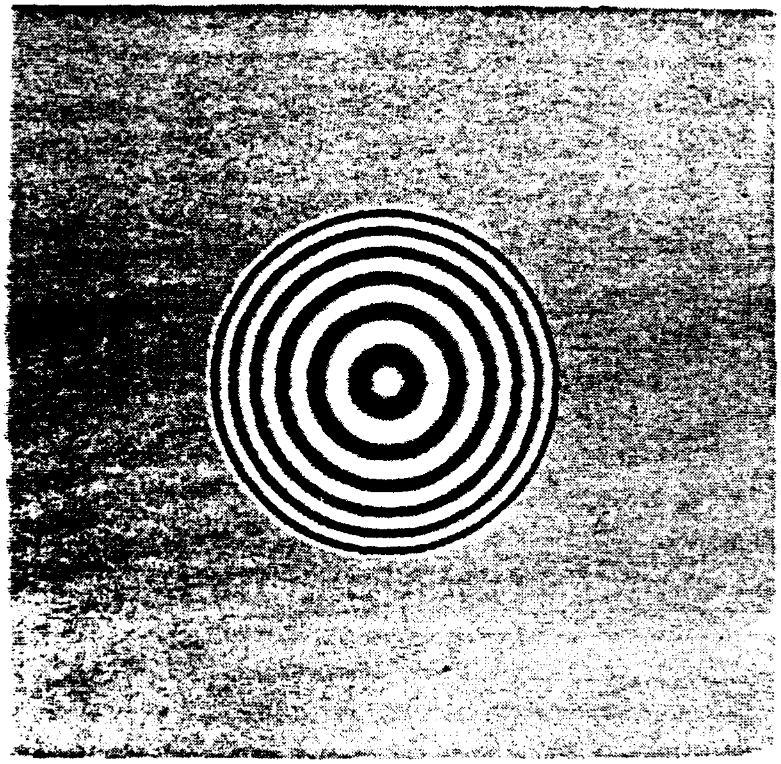


FIGURE 2.1 Radially Decaying Cosine Function

Figures 2.2 and 2.3 point out the differences between a normal hybrid transform-DPCM coder-decoder pair, figure 2.2, and the motion compensated hybrid transform-DPCM coder-decoder pair, given in figure 2.3.

Results for both pel recursive and coefficient recursive displacement estimation for a single block from the test image are contained in figures 2.4 through 2.11. The plots consist of 4 pairs of similar

These transformed blocks are then arranged into a column vector by column scanning the transformed block. They *define the* n^{th} *coefficient* of the q^{th} block of the transformed image to be,

$$c_n(q) = I^T(X_q, t)\phi_n \tag{2.6}$$

and for the estimated displaced frame from the previous image,

$$c_n(q,\hat{D}) = I^T(X_q - \hat{D}, t - \tau)\phi_n. \tag{2.7}$$

The comparable term for the pel recursive's displaced frame difference is the coefficient prediction error $e_n(q,\hat{D})$ and is given by the equation below.

$$e_n(q,\hat{D}) = [I(X_q,t) - I(X_q - \hat{D}, t - \tau)]^T\phi_n \tag{2.8}$$

Minimization with respect to the estimated displacement over the squared prediction error with a steepest descent form is given below.

$$\hat{D}_{n+1}(q) = \hat{D}_n(q) -(\varepsilon/2)\nabla_{\hat{D}_n(q)} e_n^2(q,\hat{D}_n(q)) \tag{2.9}$$

Taking the required derivative will yield the *following* iteration formula.

$$\hat{D}_{n+1}(q) = \hat{D}_n - \varepsilon e_n(q,\hat{D}_n(q))\nabla_{\hat{D}_n(q)} e_n(q,\hat{D}_n(q)) \tag{2.10}$$

Rewriting the error term gives

$$\hat{D}_{n+1}(q) = \hat{D}_n - \varepsilon e(q,\hat{D}_n(q))[(\nabla_{\hat{D}_n(q)}(I(X_q,t) - I(X_q-\hat{D}_n, t-\tau))^T\phi_n] \tag{2.11}$$

Noting that $I(X_q,t)$ is not a *function* of $\hat{D}_n(q)$ yields,

$$\hat{D}_{n+1}(q) = \hat{D}_n(q) - \varepsilon e(q,\hat{D}_n(q))[(\nabla_x I^T(X_q-\hat{D}_n,t-\tau))\phi_n] \tag{2.12}$$

When going to the next block the initial displacement estimate is set to the final estimate of the previous block.

$$\hat{D}_0(q) = \hat{D}_{N_rN_c-1}(q-1) \tag{2.13}$$

When used for motion compensated interframe hybrid transform DPCM coding, the coder transmits a quantized version of the coefficient prediction error whenever it exceeds some threshold. This allows updating the estimate of the displacement and also correcting the prediction coefficients.

This may seem to be a very strict requirement in that very little real world motion would fit the model, but on the small scale of the pixel most types of motion can be approximated by nearly pure translation. The background/foreground interaction still poses convergence problems. They next define a displaced frame difference term as is given below to represent the displacement between the actual value and its estimated value.

$$DFD(X_k,\hat{D}) = I(X_k,t) - I(X_k - \hat{D},t - \tau) \qquad (2.3)$$

Where in this case the term $\hat{D}$ is defined to be the estimated form of the displacement vector D. An attempt is made to minimize this difference through the use of a steepest descent algorithm of the form given below.

$$\hat{D}_{k+1} = \hat{D}_k - (\varepsilon/2)\nabla_{\hat{D}_k}[DFD(X_k,\hat{D}_k)]^2 \qquad (2.4)$$

Here ε is a gain term and $\nabla_{\hat{D}_k}$ is a two-dimensional gradient operator with respect to $\hat{D}_k$. Taking the required derivative in equation (2.4) will then yield the following iteration formula.

$$\hat{D}_{k+1} = \hat{D}_k - \varepsilon DFD(X_k,\hat{D}_k)\nabla_x I(X_k - \hat{D}_k,t - \tau) \qquad (2.5)$$

In this case ∇_x is the two-dimensional spatial gradient taken with respect to the row and column directions and to be evaluated at the point $X = X_k - \hat{D}$. This term, like $I(X_k - \hat{D},t - \tau)$, may require interpolating for noninteger values of $\hat{D}$ at each new iteration.

When actually used in an interframe motion compensated image coder, the transmitter will transmit any values of the $DFD(X_k,\hat{D}_k)$ term, as well as the required address information, if it exceeds some set threshold. This quantized correction is then used by the transmitter and receiver sections to update the appropriate estimates.

Following this same motion model, Netravali and Stuller [91] formulated a method for interframe image coding termed coefficient recursive estimation. It is an extension of pel-recursive with the further addition of a unitary transformation so that the operations can be carried out in the transform domain. The two methods are similar, assuming the same motion model, but now the image is partitioned into rectangular blocks of size N_r rows by N_c columns. Each element of the block is then multiplied by the appropriate transformation vector.

Ohtsuka [96] define a similar system that was actually implemented in hardware. The method used for determining the displacement vector, as before, is based on a correlational measure but now also takes into account the displacement vector of the same spatially located block from the previous frame.

Many other methods have been employed since the first frame differencing techniques. A method by Price, Snyder, and Rajala [111] is based upon a Fourier-Domain filter based on a model of the human visual system to detect motion. The model breaks up visual perception into two distinct channels. The first, their so called x channel, is a temporal low pass and spatial band pass channel. This channel carries the information contained in two-dimensional patterns with high spatial resolution but fairly low temporal dependance. They believe that it is this channel that is responsible for objects with structural complexity but with little or no motion. The other, so called y channel, is just the opposite. Its characteristics are spatial low pass and temporal bandpass. This channel, they claim, conveys the information of objects with high temporal dependance and low spatial resolution. It is this y channel that is used for the detection and analysis of motion.

2.2.3 Pel Recursive and Coefficient Recursive Displacemen

Stuller, Netravali and Robbins of Bell Laboratories start from a completely different point of view for motion. First, the end product of their work is data compression and not tracking, although it could be modified for such. The model is used for normal television data where adjacent scan rows are scanned by an interleaved method. In the first method of pel-recursive displacement estimation by Robbins and Netravali [116], the image model for pure translation with no background is given below.

$$I(X_k,t) = I(X_k - D, t - \tau) \qquad (2.2)$$

Where: $I(X_k,t)$ is the intensity of the image at the spatial location X_k and time t. $I(X_k - D, t - \tau)$. is the intensity of the image at the spatial location X_k adjusted by the displacement D at the previous time $t-\tau$. This image model is defined only for objects undergoing pure translation and not involving background or foreground interactions.

be possible as well as target motion prediction. The displacement vector need not be used strictly for motion related studies but may also be used in areas of data compression, remotely piloted vehicle control, and industrial manufacturing applications. With the detection and interpretation of motion, a very important step toward artificial vision will have been obtained.

Possible solutions of this problem that seem nontractable at present may in the near future be made possible due to the advances in VLSI technology, software development, parallel processing and electro-optical systems. So even though the process may look overly complicated and slow, there may be hope in the future.

2.2.2 Previous Work in the Field of Motion

There has been a relatively small amount of work carried out in the area of machine motion analysis until very recently when the required hardware and software became available. The work has concentrated in the areas of displacement estimation and interframe image coding. One of the earliest, and perhaps the simplest, method used for motion detection was simple image frame differencing. That is, subtract the previous frame, pixel by pixel, from the current frame and flag as motion any difference greater than some set threshold.

$$M(i,j,t) = ABS[I(i,j,t) - I(i,j,t - \tau)] \tag{2.1}$$

Motion will be defined to have occurred whenever $M(i,j,t)$ is greater than some set threshold. Although very simple, the method does show good results for a very limited class of simple images, but this method has many drawbacks that will limit its overall usefulness. First, the output is very sensitive to noise in the input images because it is a differentiating type process. Also any camera motion between image frames will translate into motion at every pixel.

There have been numerous attempts at motion compensated image coding in the past, each with its own set of advantages and disadvantages. An early method by Giorda and Racciu [30] breaks the image up into a number of small rectangular blocks and determines a displacement vector for each block via correlation. If the displacement can be found it is transmitted, if not the entire block must be transmitted. Ninomiya and previous frame and if possible transmit the change information as a function of motion. Even though the requirements can be simply stated the actual implementation has proved to be difficult.

2.2.1 Displacement and Motion

Simply stated, motion is defined to be a time series of displacements. That is, in order for motion to be perceived, time must pass and a displacement must take place. If artificial vision and intelligence is ever to become a reality, then a sufficiently good model for motion will have to be employed. For this reason, and the fact that memory space is always limited, the vision system for motion analysis should be based on some time adaptive displacement algorithm.

The problem can now be stated: Find a method to locate the spatial displacement from one image frame to the next, such that an estimate of the direction and magnitude of any detected motion can be made. This estimate should be based upon the current frame, previous frame and previous displacement vectors.

The problem statement is simple enough, but the effort is complicated by many other external factors. One of these factors involves object/background and object/foreground interaction. For example, if an object in the input frame moves in such a way so as to uncover some new background, complications will arise in that the new information from frame to frame now consists not only of the information contained in the object motion, but also in the new background that is uncovered. The imaging system should have the capability to detect the difference between the moving target and the nonmoving uncovered background. Another problem simpler in scope than the above, but just as important, involves the loss of moving objects from the field of view and the addition of new moving objects into the field of view. Again the system should be able to detect and track these new moving objects and discard those that exit the field of view. Finally, objects moving in the field of view may become partially or totally hidden by both moving and nonmoving objects and the system should be capable of adapting to this.

Once the motion has been identified, there can be many uses for the information thus provided. Visual tracking of moving objects will then

2.1.4 Other Methods

The above mentioned methods are just a few of the many possibilities that have been proposed. Other methods include normal pulse code modulation (PCM) as well as adaptive forms of PCM. There are methods based on the statistics and histogram of images that code the most common gray levels into short code words and rare gray levels into long code words.

Hybrid coding is a combination of both transform and predictive coding and lately has been studied quite extensively [36], [37], [54]. The references contain many other variations and combinations of these and many more methods.

2.2 METHODS EMPLOYING MOTION COMPENSATION

Up to this point most of the mentioned coding schemes have been used for intraframe image coding, with many being extended to include image sequences or inter-frame image coding. The inclusion of the third dimension, temporal or time, allows for the exploitation of the correlation that exists in this temporal direction.

The human visual system is a very fine imaging system and very susceptible to many kinds of image degradation. The degree to which certain degradations affect the imaging process vary with respect to the kind of degradation. It has been shown that the eye is very critical about detail in still pictures but becomes less critical if the scene contains motion. If the entire scene is undergoing a slow constant translation the amount of detail required remains high. With these simple requirements in mind, it can be seen that any inter-frame coding system with a human as the final viewer will be required to maintain a high degree of detail in areas of little or no motion and also during times when the camera itself moves slowly as in panning. The areas that may be somewhat degraded by the coding system with little loss of information to the human visual system are the areas in the images undergoing translation.

The area of motion compensated image coding lends itself directly to these requirements. The premise of motion compensated image coding is a simple one, transmit only the portions of the image that differ from the

the predictive techniques can not offer. Because of the rather high correlation in most images, the energy tends to be concentrated in the lower frequency or sequency coefficients. Hence, these transform coefficients can then be transmitted if their energy content exceeds some value. If the value does not exceed this threshold then the coefficient may be grossly quantized or not transmitted at all. Habibi and Wintz [35] combined this method with block quantization to obtain good quality results on the order of 2 bits/pixel when the original image was coded with 8, or a data reduction by a factor of 4. Ngan [95] extended the normal transform coding with the addition of adaptation to both the quantization and bit selection.

There are many different unitary transforms that can be used for transform domain image coding. Some of the more popular are the Hadamard, Fourier, Slant, Cosine, Sine and Karhunen-Loeve. Each method packs the energy differently and different results will be obtained if the discard method remains the same. Hideo Kitajima [62] has provided a method for determining the energy packing efficiency of one of the more useful transforms, the Hadamard transform.

2.1.3 Predictive Coding

Another widely used coding technique for digital images is what is generally termed predictive coding [19],[23],[36],[37],[38],[55],[92], and [122]. The predictive coders differ from the above two methods in that data is not just discarded. As the name implies, a prediction is made by using local pixel intensity values in some predescribed combination and then quantizing and transmitting the error. Because of the high degree of correlation normally occurring between adjacent pixels the prediction process works quite well, which implies that the error is normally much smaller than the original signal. This smaller error signal, in terms of the mean square power, is what enables the predictive coder to obtain its high degree of data compression. The more structured and correlated the image, the better the predictor is able to function and hence the lower the data rate for a fixed fidelity criterion.

2.1 PREVIOUS METHODS FOR IMAGE CODING

There is a wealth of information available on the topic of general data coding and picture coding in the literature. Many different methods, applications and theories are available from a large number of authors. A review of many of coding solutions is available from Netravali and Limb [92], Habibi [35]-[38], Pratt [110], Castleman [14], Gonzales and Wintz [31], and Andrews and Hunt [4]. Some of the more important methods will be briefly described in the following sections.

2.1.1 Subsampling

One of the simplest forms of data compression for digital images involves subsampling the source image and interpolation of the discarded points at the receiver. This can be accomplished in two different ways. The first involves using only the upper right hand pixel value of each of the N by N sub-blocks and transmitting that value to represent the entire sub-block. The data rate here, as in the second method, is only $1/N^2$ of the original data rate. The second method is similar to the first but transmits the average of the sub-block instead of the upper right hand pixel. As would be expected, these methods tend to discard much of the information in order to decrease the data rate. If the image was already sampled above the Nyquist rate, then some data reduction could take place with little loss to the quality of the output. Normally the image is sampled less than the Nyquist rate and any subsampling will cause a substantial information loss.

2.1.2 Transform Coding

The simple method of discarding data has been shown not to function adequately for most images, but with the inclusion of a unitary transformation data can be discarded with much less loss in image quality. The function of the unitary transform is to redistribute the image intensity energy in such a way that it is mostly contained in a small number of transform coefficients. It does this by decorrelating the spatial data and hence minimizing the statistical redundancy of the information to be coded. Because the coding is done in parts or sections, an error in one part will not propagate to other sections if, for example, a channel error is made. This is an advantage that many of

Chapter II

MOTION COMPENSATED IMAGE CODING AND DISPLACEMENT ESTIMATION

In chapter one some of the requirements and restrictions that necessitate data compression as well as some possible applications for its use were presented. In this chapter the basic principles of data compression will be reviewed and some examples of various methods will be presented. Only a few of the many methods proposed in the past will be presented due to the very large number of papers written on the subject.

The purpose of data compression is to take a data sequence, be it image data, speech or any other information source, and perform a data manipulation in such a way so as to be able to reproduce the original signal with an acceptable amount of degradation using a smaller bandwidth. The degree to which this goal is reached is based both upon the statistics and form of the image as well as the operation of the coding system itself. This data compression is often achieved through removal of the large amount of interpixel correlation that is normally present in most data.

One of the early uses of data compression was in the field of digital voice communication. The signal generated by the human vocal system tends to be quite correlated and hence various predictive techniques have been used to obtain good compression results. When the field of digital image coding opened, many of the methods used for speech were directly adapted for use with the digital images. These procedures were not able to take full advantage of the geometrical, statistical or cognitive structure of the image data. Voice, being a single dimensional signal, does not have the geometrical structure or the two-dimensional statistical dependence that image data has. The optimal compression scheme should be able to take full advantage of any structure that the actual image has to offer. In most cases only a small subset of this structure is actually exploited.

difficult and expensive to put up a new satellite or add the required hardware on a deep space probe. Military applications can include remotely piloted vehicles, remote surveillance and other imagery applications. One side effect of data compression that often times is important in military applications is that of secure data transmission. If the transmitter and receiver are not a matched pair, the receiver will not be able to recombine the data sequence to obtain the original image. There are many different ways to achieve this bandwidth reduction and most, if not all, exploit the high correlation between adjacent pixels to reduce the data rate.

In chapter two previous work in the field of image coding is presented. It provides a wide overview of the many different methods that have been tested and employed in the past for bandwidth compression or reduction. The emphasis of chapter two is in the area of motion compensated image coding. This is coding that exploits knowledge of the motion between frames in sequential image data. In chapter three a new method for motion compensated image coding, namely Prediction Coefficient Energy Concentration is presented. The derivation of the displacement estimation procedure as well as the coding procedure is presented. In chapter four information on the implementation and testing of the Prediction Coefficient Energy Concentration model is provided. It presents the information that ties the theoretical aspects of the algorithm to its actual simulation and modelling. The results of the model as represented by a software implementation are presented in chapter five. Actual output images as well as the analysis of the algorithm is presented.

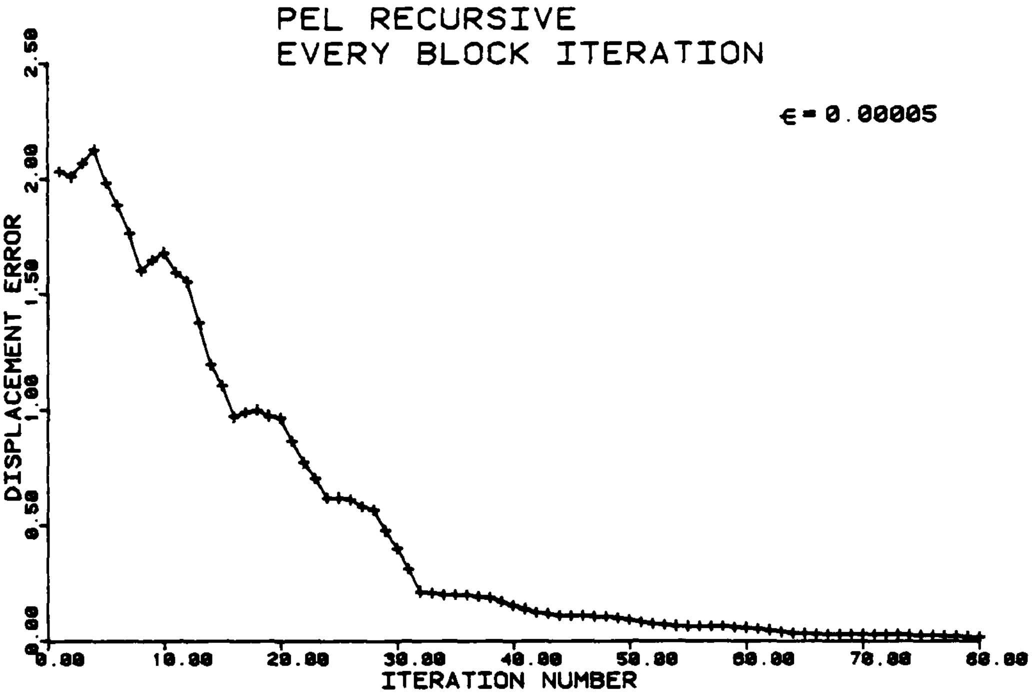


FIGURE 2.4 Modified Pel Recursive Iteration Graph

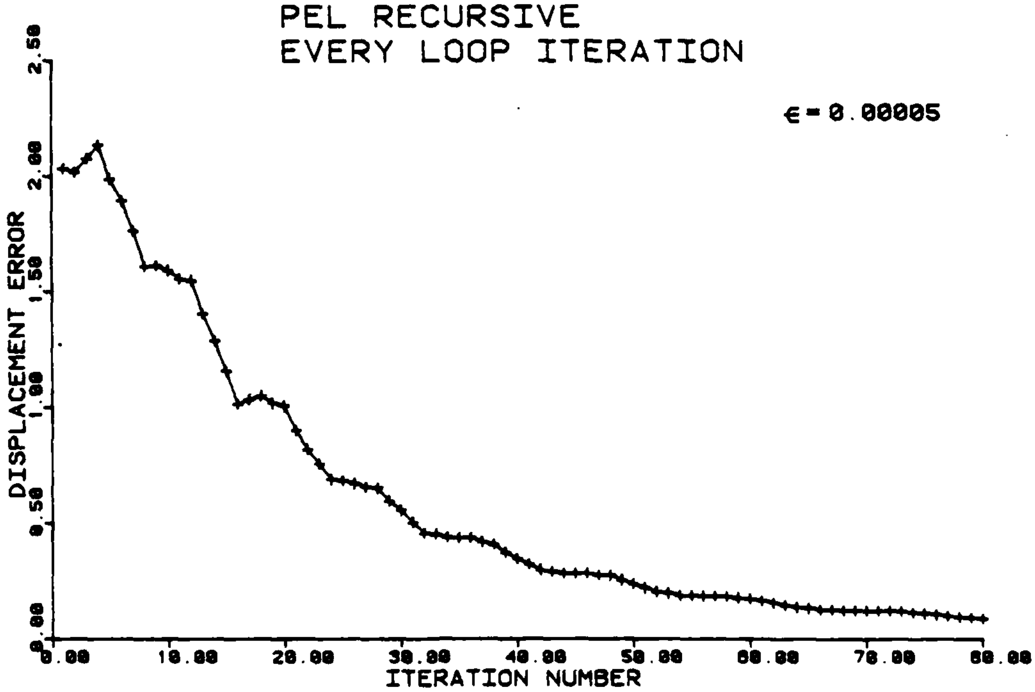


FIGURE 2.5 Pel Recursive Iteration Graph

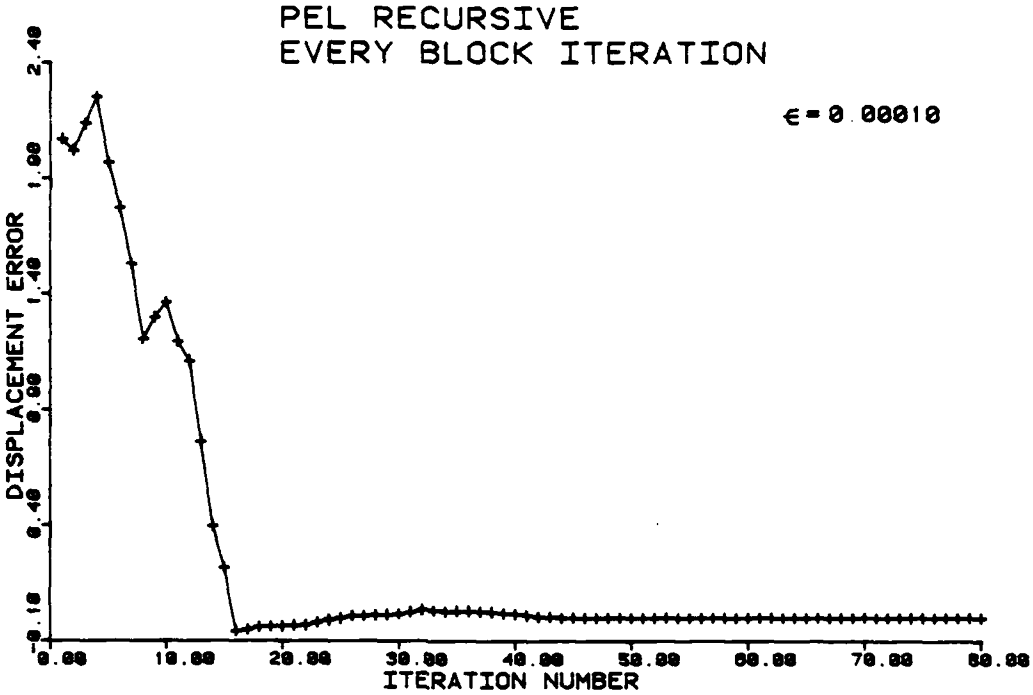


FIGURE 2.6 Modified Pel Recursive Iteration Graph

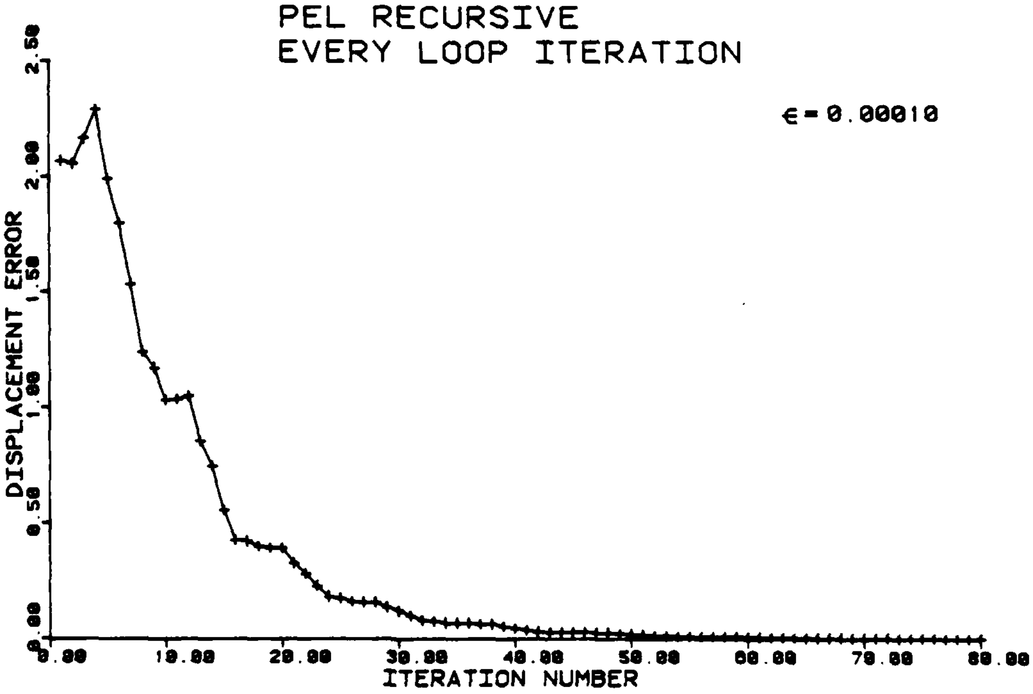


FIGURE 2.7 Pel Recursive Iteration Graph

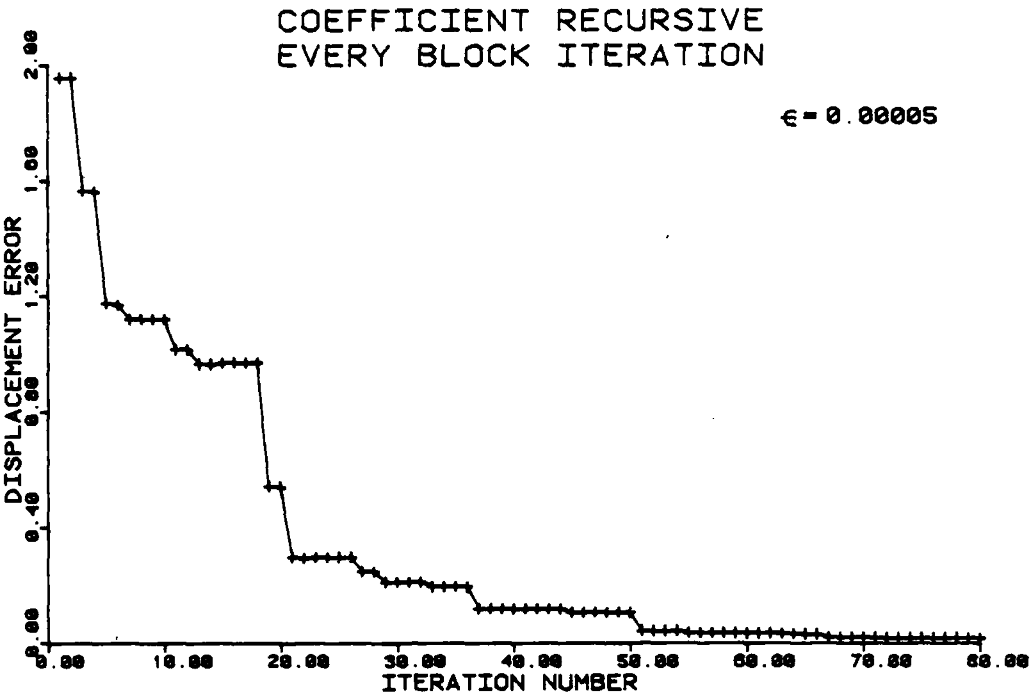


FIGURE 2.8 Modified Coefficient Recursive Iteration Graph

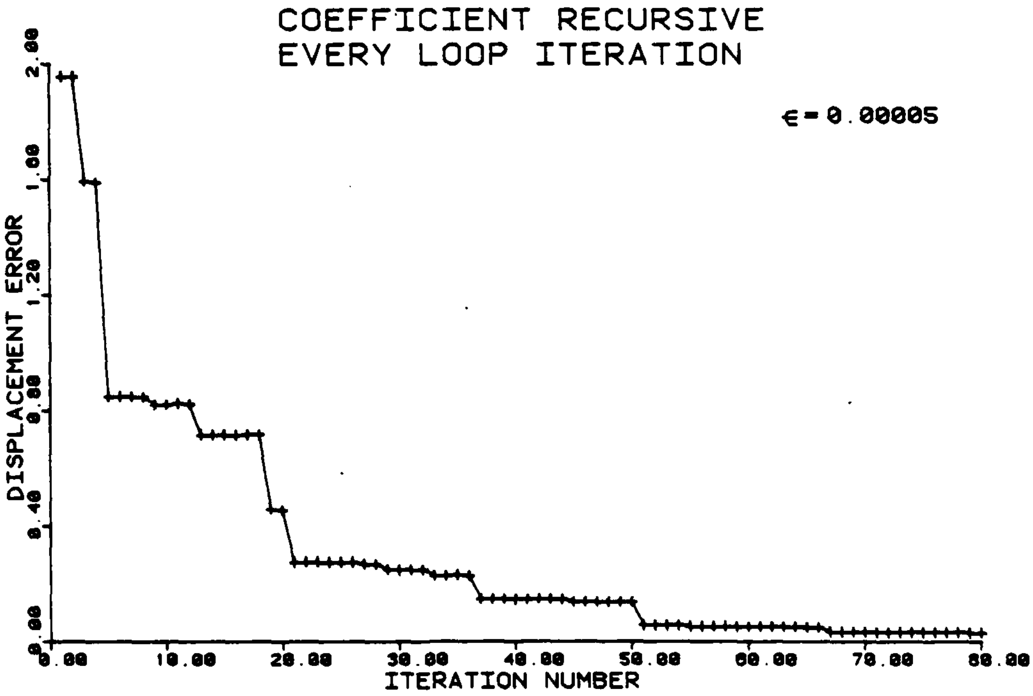


FIGURE 2.9 Coefficient Recursive Iteration Graph

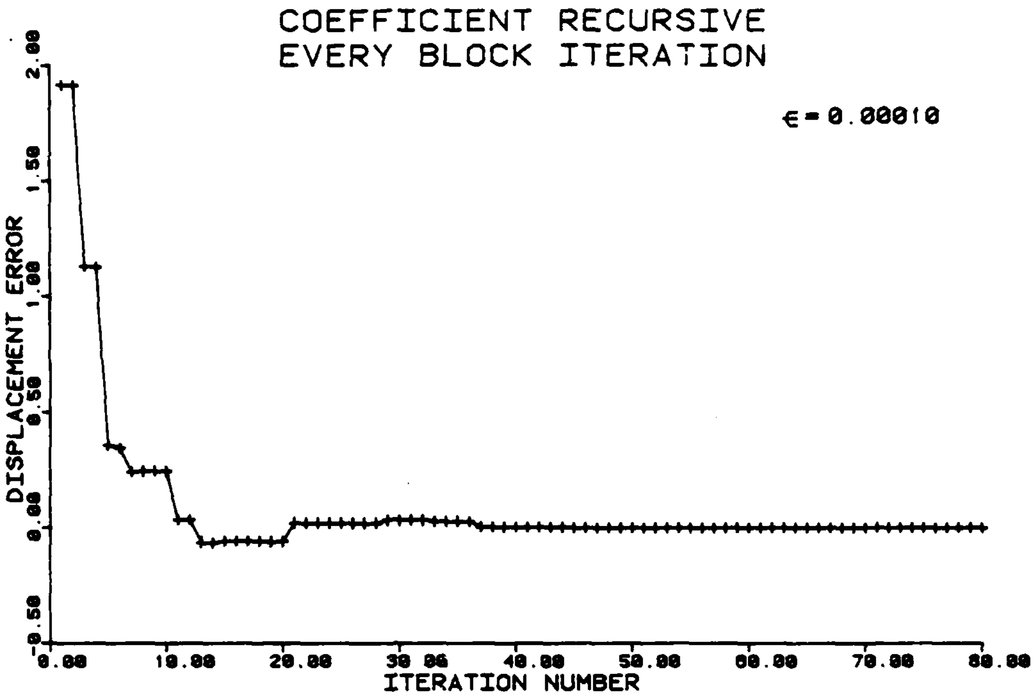


FIGURE 2.10 Modified Coefficient Recursive Iteration Graph

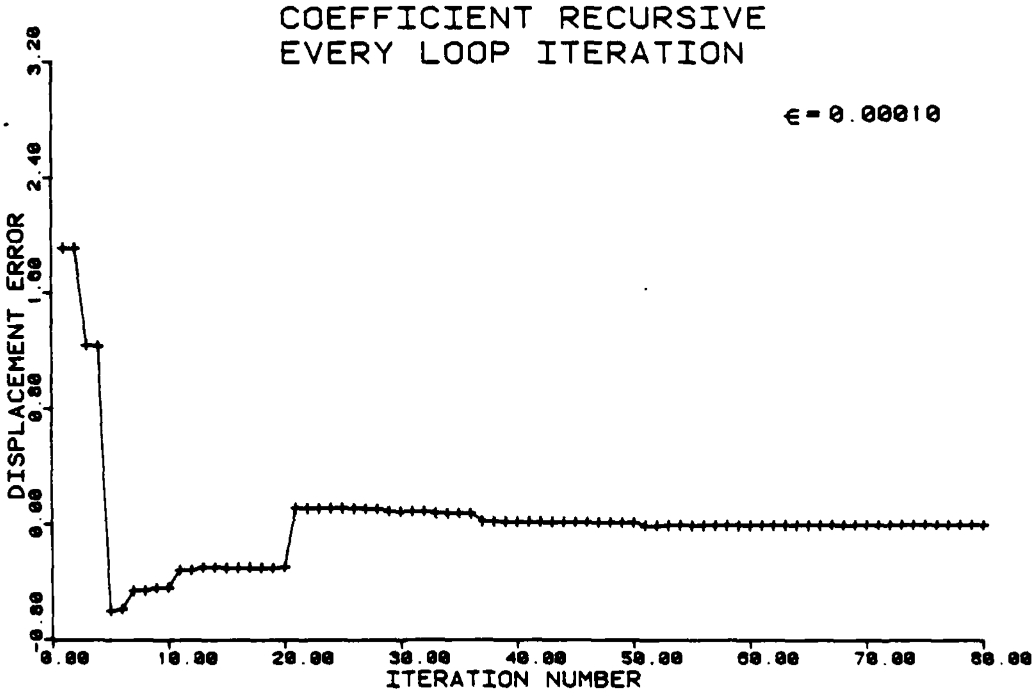


FIGURE 2.11 Coefficient Recursive Iteration Graph

2.2.4 Residual Recursive

The method of Residual Recursive by Rashid and Jones [112], [113] is similar to both the pel recursive and coefficient recursive methods of the last section. This method is based upon an earlier procedure developed by Jones [54], [55] for image coding, namely Adaptive Hybrid Picture Coding or AHPC for short. The advantages offered by the combination of AHPC and the recursive algorithms developed by Stuller, Netravali, and Robbins lie in the determination of the gradient term.

Before getting to the motion related derivation, it is necessary to gain some background information on AHPC. AHPC is a data reduction coding method used for intraframe data compression. The method is based upon a two step correlation removal technique. The first stage of the process involves a one dimensional unitary transformation in the column direction to reduce the columnwise correlation. The row correlation is then reduced through a prediction process that operates in the row direction. Data compression is achieved because the mean square power of the prediction error and predictor overhead is less than the mean square power of the original. The prediction error will be shown to approximate the innovations process for the image data and hence the image gradient. A block diagram for the AHPC process is provided in figure 2.12.

The transformed pixels are modelled as a one dimensional autoregressive series where the predicted value of the signal is defined to be an optimally weighted linear combination of previously reconstructed transformed picture elements r_n. The weighting factors are the predictor coefficients and hence the prediction equations can be written as follows.

$$r_n = \sum_{k=1}^{p} a_k r_{n-k} \qquad (2.15)$$

The difference or error signal e_n is then defined to be the difference between the original signal s_n and the predicted value r_n.

$$e_n = s_n - r_n \qquad (2.16)$$

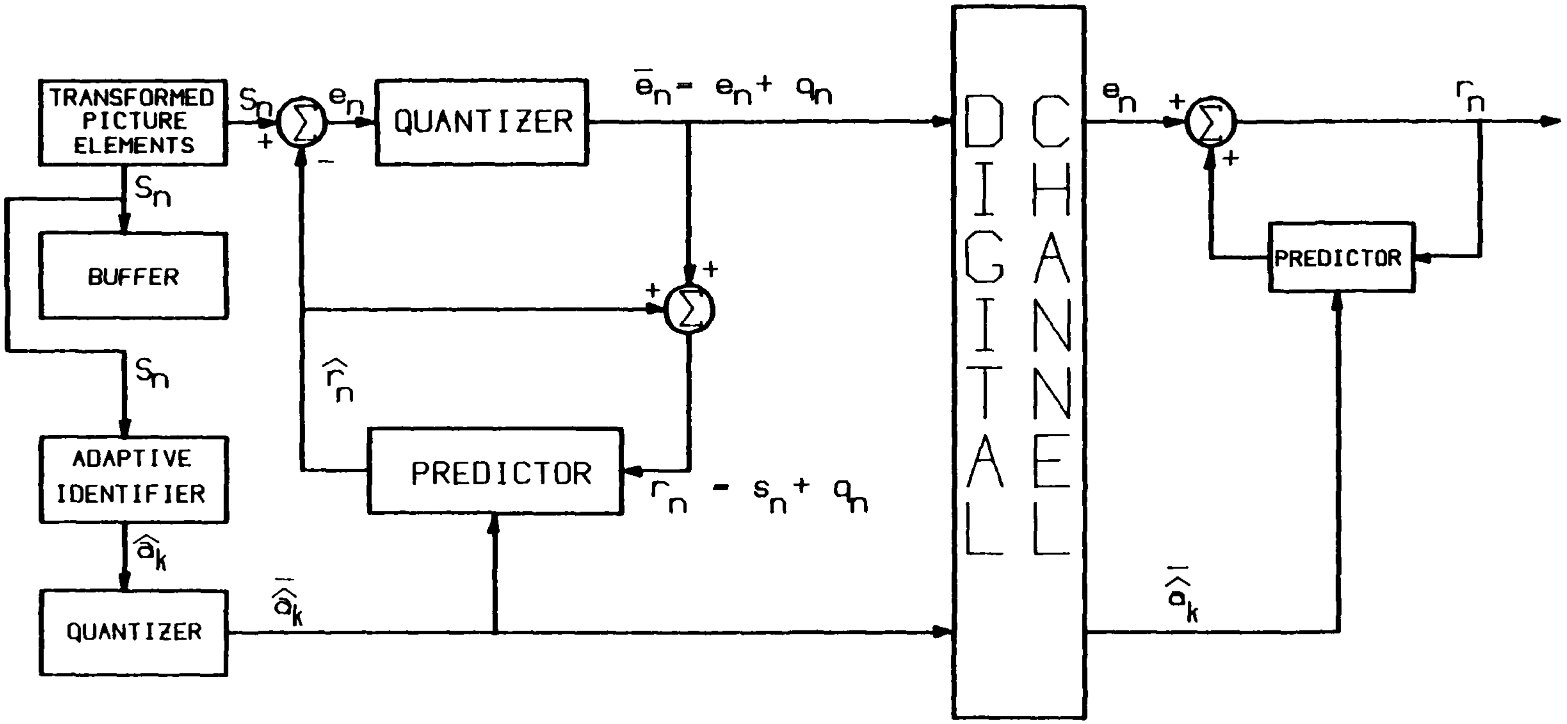


FIGURE 2.12 AHPC Block Diagram

Using the mean square error as the optimality criterion, the value for the predictor coefficients are chosen such that P_e, the mean square error, is minimized and given below.

$$P_e = \frac{1}{L} \sum_{m=1}^{L} e_m^2 \tag{2.17}$$

Because of the adaptability of the algorithm, the predictor coefficients a_k remain constant for a single row or learning period of length L. Due to the statistical change of the data from row to row, the predictor coefficients must also change at row boundaries. Hence, the optimal error signal for a single learning period can be written as below.

$$e_n = s_n - A_n^T R_n \tag{2.18}$$

The sample value n is allowed to vary from 1 to the learning period L. A_n^T is a column vector of the optimal set of predictor coefficients and R_n is a row vector of the p previous reconstructed transformed picture elements.

As stated earlier, the optimal predictor vector is chosen to minimize the average error signal power and amounts to minimizing the following variance term.

$$VAR(e) = \frac{1}{L}[VAR(s) - COV(s,r)A_n - A_n^T COV(r,s) + A_n^T VAR(r)A_n] \tag{2.19}$$

The minimization is accomplished by taking the partial derivative with respect to A_n.

$$A_n(opt) = dVAR(e)/dA_n \tag{2.20}$$

This yields,

$$A_n(opt) = VAR(r)^{-1} COV(s) \tag{2.21}$$

When complete statistical knowledge of the source is not known, a method is needed to produce good estimates of the statistics during each learning period. If the predictor vector is treated as a state and a linear minimum error variance sequential estimate of the state is desired, then sequential estimation theory can be used. In this case Kalman filtering is used.

Given the optimal set of predictor coefficients, the reconstructed transformed picture elements can be generated from the following equation.

$$s_n = \bar{A}_p^T R_n(p) + e_n \qquad (2.22)$$

Here the p signifies the predictor length, $R_n(p)$ the p previous reconstructed transformed picture elements and $\bar{A}_n^T$ is the quantized identified predictor vector of length p. The identification filter algorithm from [54] can then be written

$$\hat{A}(k+1|S(k+1)) = [I - K(k+1)R_p^T(k)]\hat{A}(k|S(k)) + K(k+1)s(k). \qquad (2.23)$$

The difference here between s(k) and S(k) is that s(k) is a single observation while S(k) is all observations up to and including s(k). Hence the desired identification algorithm becomes,

$$A(k+1) = A(k) + K(k+1)[s(k) - R_p^T(k)A(k)] \qquad (2.24)$$

where the bracketed term is simply the prediction error. The gain term K(k+1) is given by,

$$K(k+1) = V_A(k)R_p(k)/(V_z + R_p^T(k)V_A(k)R_p(k)). \qquad (2.25)$$

$V_A(k)$ is simply the variance of the identification error of A(k) as given below.

$$A(k) = A(k) - \hat{A}(k) \qquad (2.26)$$

V_z is a scalar term not obtainable recursively and is set to a constant value. Given the background for AHPC, its relationship to motion compensated image coding can be discussed. The image motion model for both pel-recursive and coefficient recursive displacement estimation is

$$I(X_k, t) = I(X_k - D, t - \tau) \qquad (2.27)$$

as was originally given in equation (2.2). In the transform domain the equivalent coefficients are given below as originally in (2.6).

$$c_n(q) = I^T(X_q, t)\phi_n \qquad (2.28)$$

Note the values given for the displaced frame term as in equation (2.7). The method of residual recursive displacement estimation lends itself more closely to coefficient recursive displacement estimation and hence will be compared as such. Note first that the transform domain coefficients of the coefficient recursive method are also the signal samples from AHPC.

$$c_n(q) = s_n(q) \tag{2.29}$$

That is to say, the n^{th} coefficient of the q^{th} block is the same in both notations provided the linear unitary transform is the same. Hence, the displaced terms then become,

$$c_n(q,D) = s_n(q,D) \tag{2.30}$$

Replacing this notation in equation (2.17) yields

$$e_n(q) = c_n(q) - A_n^T\hat{C}_n(q). \tag{2.31}$$

Where, as before, the capital letter $\hat{C}_n(q)$ defines a vector of the p previous predicted values from the q^{th} block. On a coefficient-by-coefficient basis, this can be written

$$e_n(q) = c_n(q) - \sum_{k=1}^{p} a_k\hat{c}_{n-k}(q). \tag{2.32}$$

Likewise the displaced term becomes

$$e_n(q,\hat{D}) = c_n(q,\hat{D}) - \sum_{k=1}^{p} a_k(\hat{D})\hat{c}_{n-k}(q,\hat{D}) \tag{2.33}$$

Note that the predictor values $a_k(\hat{D})$ are now functions also of the estimated displacement $\hat{D}$. Recall that the coefficient prediction error, now with the prime added, is

$$e'_n(q,\hat{D}) = [I(X_q,t) - I(X_q - \hat{D},t - \tau)]^T\phi_n. \tag{2.34}$$

Which can also be written as,

$$e'_n(q,\hat{D}) = c_n(q) - c_n(q,\hat{D}). \tag{2.35}$$

The comparable term in residual recursive displacement estimation is the displaced residual difference or $DRD_n(q,\hat{D})$

$$DRD_n(q,\hat{D}) = e_n(q) - e_n(q,\hat{D}). \tag{2.36}$$

As before, a steepest descent algorithm is used to minimize the squared difference with respect to $\hat{D}_n$. The form of the algorithm is given below.

$$\hat{D}_{n+1} = \hat{D}_n -(\varepsilon/2)\nabla_{\hat{D}_n} [DRD_n(q,\hat{D}_n)]^2 \tag{2.37}$$

Taking the indicated gradient results in

$$\hat{D}_{n+1} = \hat{D}_n - \varepsilon DRD_n(q,\hat{D}_n)\nabla_{\hat{D}_n} DRD_n(q,\hat{D}_n) \tag{2.38}$$

Before determining the gradient of the displaced residual difference term, it is necessary to rewrite it in its constituent elemental form. From (2.32),(2.33) and (2.36)

$$DRD_n(q,\hat{D}_n) = c_n(q) - \sum_{k=1}^{p} a_k\hat{c}_{n-k}(q) - c_n(q,\hat{D}_n) + \sum_{k=1}^{p} a_k(\hat{D}_n)\hat{c}_{n-k}(q,\hat{D}_n) \tag{2.39}$$

The $c_{n-k}(q)$ is the reconstructed version of $c_{n-k}(q)$ Noting equation (2.35)

$$DRD_n(q,\hat{D}_n) = e'(q,\hat{D}_n) - \sum_{k=1}^{p} a_k\hat{c}_{n-k}(q) + \sum_{k=1}^{p} a_k(\hat{D}_n)\hat{c}_{n-k}(q,\hat{D}_n) \tag{2.40}$$

Rewriting using only a single summation

$$DRD_n(q,\hat{D}_n) = e'(q,\hat{D}_n) - \sum_{k=1}^{p} a_k\hat{c}_{n-k}(q) - a_k(\hat{D}_n)\hat{c}_{n-k}(q,\hat{D}_n) \tag{2.41}$$

At this point it is necessary to make use of the following identity.

$$\sum_{i=1}^{n} a_iA_i - \sum_{i=1}^{n} b_iB_i = \sum_{i=1}^{n} c_i(A_i - B_i) \tag{2.42}$$

That is given a, b, A, and B there exists a set of c's that satisfy the above equation. In the most strict sense, if they pair up on a point by point basis then the following equation will hold.

$$a_iA_i - b_iB_i = c_i(A_i - B_i) \tag{2.43}$$

In this case,

$$c_i = [(a_iA_i) - (b_iB_i)]/(A_i - B_i) \tag{2.44}$$

Except for the possible case where $A_i = B_i$, in which case c_i will equal zero. So with this, there exists a set of coefficients b_k's that will satisfy the following.

$$DRD_n(q,\hat{D}_n) = e'_n(q,\hat{D}_n) - \sum_{k=1}^{p} b_k(\hat{D}_n)[\hat{c}_{n-k}(q) - \hat{c}_{n-k}(q,\hat{D}_n)] \tag{2.45}$$

Because of the good prediction and error quantization, a simplifying assumption can be made in that c_n is equal to $\hat{c}_n$ or

$$c_n = \hat{c}_n \tag{2.46}$$

and

$$c_n(q,\hat{D}_n) = \hat{c}_n(q,\hat{D}_n). \tag{2.47}$$

With this equation, (2.45) becomes

$$DRD_n(q,\hat{D}_n) = e'_n(q,\hat{D}_n) - \sum_{k=1}^{p} b_k(\hat{D}_n)[c_{n-k}(q) - c_{n-k}(q,\hat{D}_n)] \tag{2.48}$$

Using equation (2.35) yields,

$$DRD_n(q,\hat{D}_n) = e'_n(q,\hat{D}_n) - \sum_{k=1}^{p} b_k e'_{n-k}(q,\hat{D}_n) \tag{2.49}$$

This says that the displaced residual term is composed of the current residual minus a linear combination of previous residuals or errors. Combining (2.39) with (2.46) and (2.47) yields,

$$DRD_n(q,\hat{D}_n) = c_n(q) - c_n(q,\hat{D}_n) - \sum_{k=1}^{p} a_k c_{n-k}(q) + \sum_{k=1}^{p} a_k(\hat{D}_n) c_{n-k}(q,\hat{D}_n) \tag{2.50}$$

Taking the derivative of $DRD_n(q,\hat{D}_n)$ with respect to $\hat{D}_n$ will show that the first and third terms on the right side of equation (2.50) will go to zero. These are not functions of $\hat{D}_n$. The remaining gradient term is then,

$$\nabla_{\hat{D}_n} DRD_n(q,\hat{D}_n) = -\nabla_{\hat{D}_n} c_n(q,\hat{D}_n) + \nabla_{\hat{D}_n} \sum_{k=1}^{p} a_k(\hat{D}_n) c_{n-k}(q,\hat{D}_n) \tag{2.51}$$

This is the most general form of the gradient equation, note that the first term on the right hand side of (2.51) is what is termed the coefficient gradient vector in coefficient recursive displacement estimation. It is this term that is very closely approximated by the AHPC residual or error signal. That is,

$$e_n(q,\hat{D}_n) = -\nabla_{\hat{D}_n} c_n(q,\hat{D}_n) \tag{2.52}$$

and hence it is not required to calculate the coefficient gradient at every iteration. Replacing this in equation (2.51) will yield,

$$\nabla_{\hat{D}_n} DRD_n(q,\hat{D}_n) = e_n(q,\hat{D}_n) + \nabla_{\hat{D}_n} \sum_{k=1}^{p} a_k(\hat{D}_n) c_{n-k}(q,\hat{D}_n). \tag{2.53}$$

The gradient is equal to the AHPC residual plus the gradient of the prediction term. Neglecting the second term, the form is similar to that of coefficient recursive displacement estimation. Using equation (2.53) in equation (2.38) will yield the following iteration formula.

$$\hat{D}_{n+1} = \hat{D}_n - \varepsilon DRD_n(q,\hat{D}_n)[e_n(q,\hat{D}_n) + \nabla_{\hat{D}_n} \sum_{k=1}^{p} a_k(\hat{D}_n) c_{n-k}(q,\hat{D}_n)] \tag{2.54}$$

If the last term in (2.54) is neglected, the following iteration formula is very close to the form used in equation (2.12).

$$\hat{D}_{n+1} = \hat{D}_n - \varepsilon DRD_n(q,\hat{D}_n) e_n(q,\hat{D}_n) \tag{2.55}$$

Another approach that used all of the available information can be obtained from the displaced residual difference term as given in equation (2.36) when combined with (2.52) to yield the following.

$$DRD_n(q,\hat{D}_n) = -\nabla_{\hat{D}_n} c_n(q,\hat{D}_n) \tag{2.56}$$

The gradient of this term then becomes,

$$\nabla_{\hat{D}_n} DRD_n(q,\hat{D}_n) = -\nabla^2_{\hat{D}_n} c_n(q,\hat{D}_n) \tag{2.57}$$

and the iteration formula can hence be written as,

$$\hat{D}_{n+1} = \hat{D}_n - \varepsilon \nabla_{\hat{D}_n} c_n(q,\hat{D}_n) \nabla^2_{\hat{D}_n} c_n(q,\hat{D}_n) \tag{2.58}$$

or,

$$\hat{D}_{n+1} = \hat{D}_n - \varepsilon DRD_n(q,\hat{D}_n) \nabla^2_{\hat{D}_n} c_n(q,\hat{D}_n). \tag{2.59}$$

Results for this method are given in figures 2.14 through 2.17 for the same image data as was used for the earlier methods. The FORTRAN program used to test the algorithm and generate the plots is contained in appendix A.

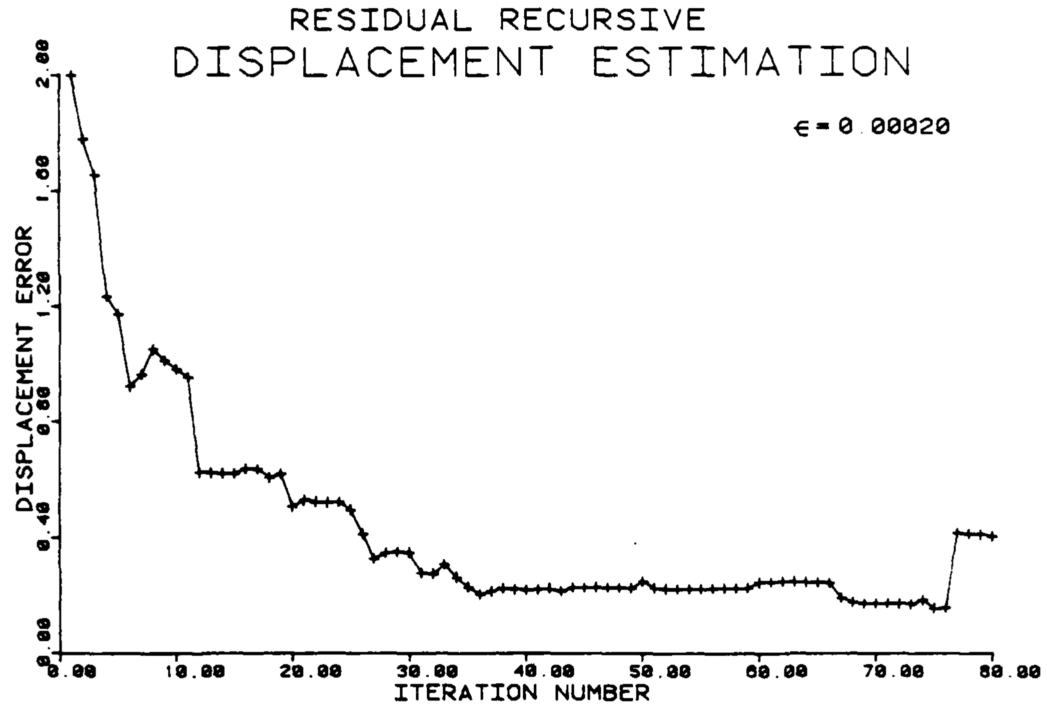


FIGURE 2.13 Residual Recursive Displacement Estimation

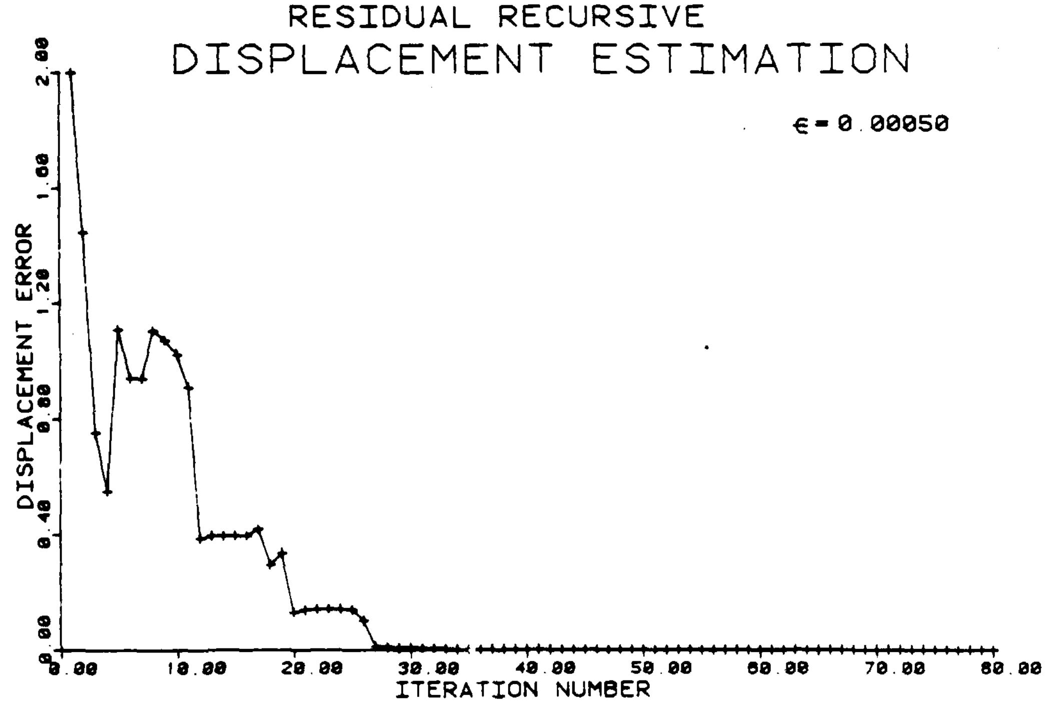


FIGURE 2.14 Residual Recursive Displacement Estimation

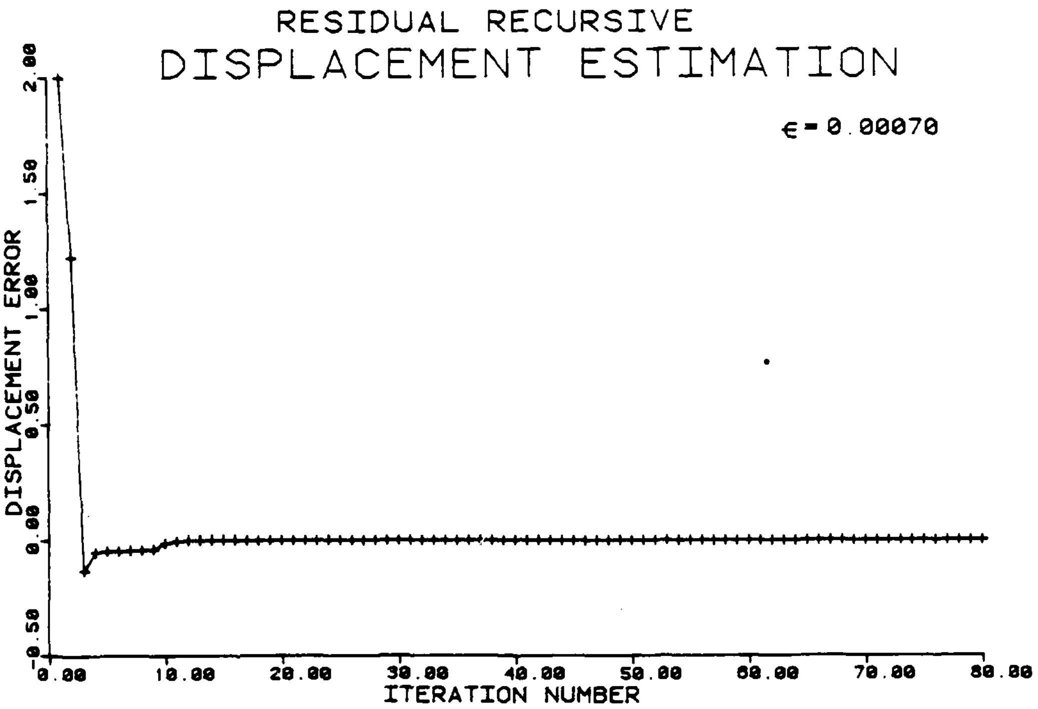


FIGURE 2.15 Residual Recursive Displacement Estimation

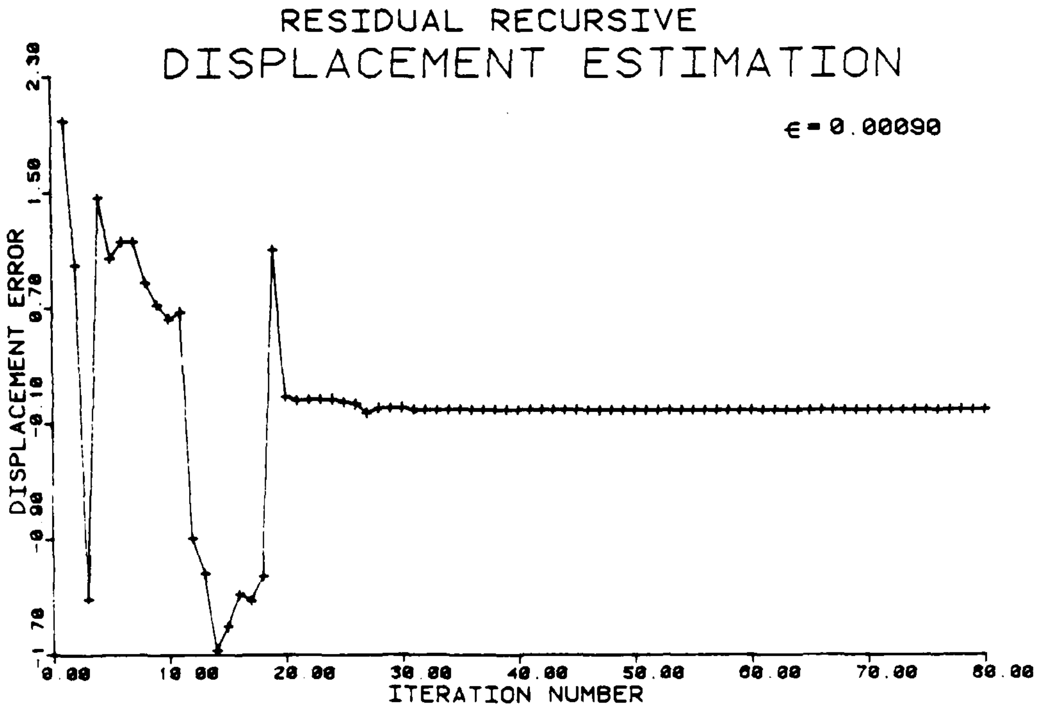


FIGURE 2.16 Residual Recursive Displacement Estimation

function of the data. Here, as in the two previous methods, the minimum value of the metric or cost function determines the nearest integer displacement. This is the method that was actually employed for the results presented in the later sections.

3.4 DISPLACEMENT QUADRANT ESTIMATION

Given an estimate for the integer displacement, the next problem is then to determine if this estimate is high or low in both the X and Y directions. In other words, the displacement needs to be bounded to an integer displacement cell. The problem can be better visualized by looking at the four displacement cells, or quadrants, about the integer estimate as is shown in figure 3.5.

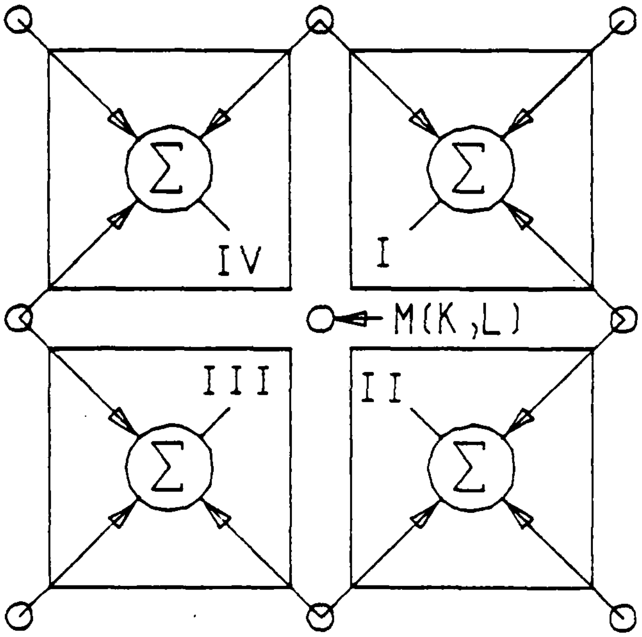


FIGURE 3.5 Method for Minimum Quadrant Generation

The quadrants are numbered I through IV starting with the upper right hand corner and moving in a clockwise direction. Each quadrant has associated with it four values of the metric, one being the minimum value that defined the integer displacement estimate. Using the values generated by the chosen metric, it is possible to determine which quadrant is minimum. This is accomplished by summing the values of the

Where $\hat{D}$ is defined to be an integer displacement, I is the actual intensity value of the current frame, while $I(\hat{D})$ is the current frame estimate conditioned on the integer valued displacement vector $\hat{D}$. Therefore,

$$R = \sum_{I(D)} \sum_{I} C(I(\hat{D}),I)P_{i(\hat{D}),i}(I(\hat{D}),I)). \qquad (3.29)$$

The joint probability on I and $I(\hat{D})$ is defined as $P_{i(\hat{D}),i}(I(\hat{D}),I)$. Rewriting the joint density in terms of a conditional density yields,

$$R = \sum_{I(\hat{D})} \sum_{I} C(I(\hat{D}),I)P_{i|i(\hat{D})}(I|I(\hat{D}))P_{i(\hat{D})}(I(\hat{D})). \qquad (3.30)$$

All of the terms $C(I(\hat{D}),I)$, $P_{i|i(\hat{D})}(I|I(\hat{D}))$ and $P_{i(\hat{D})}(I(\hat{D}))$ are non-negative, so R can be minimized by minimizing the inner sum.

The conditional distribution is unknown. It is therefore assumed to be uniform. This is equivalent to discounting the effects of the statistical properties of the image sequence. With this assumption, it is required to minimize

$$R = \sum_{I} C(I(\hat{D}),I). \qquad (3.31)$$

This is accomplished by determining the values of the cost function over a -p to p neighborhood in both spatial directions with respect to the previous frame. Further, a two valued threshold function needs to be defined for the cost function.

$$C(a,b) = 0 \text{ if } b<a \qquad (3.32)$$

$$C(a,b) = 1 \text{ if } b>a \qquad (3.33)$$

The choice of this metric produces a uniform cost function. When this is combined with the absolute value method, it yields a counting arrangement as follows.

$$M(i,j) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} C(T,ABS[I(m,n,t) - I(m+i,n+j,t-\tau)]) \qquad (3.34)$$

The value chosen for the threshold T is fairly arbitrary, but somewhat loosely related to the variance of the noise and may even itself be a

3.3.1 Sum of Absolute Values of Difference Method

The first method is based on the sum of the absolute values of the difference term.

$$M(i,j) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} ABS[I(m,n,t) - I(m+i,n+j,t-\tau)] \tag{3.26}$$

Where both i and j are allowed to vary from -p to p. ABS in the above implies taking of the absolute value of the quantity in the square brackets. The best integer displacement is then defined to occur at the point at which the metric M(i,j) is minimum. From a cost function point of view, the absolute value function defines an absolute error cost function.

3.3.2 Sum of Squares of Difference Method

The method is similarly defined as that in (3.26) with the use of a squaring function instead of the absolute value. This offers the advantage of penalizing the metric greater for large differences than for a number of smaller differences.

$$M(i,j) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} [I(m,n,t) - I(m+i,n+j,t-\tau)]^2 \tag{3.27}$$

Here again the minimum value of the metric defines the location of the best integer estimate. From a cost function point of view, the squaring function defines a square-error cost function.

3.3.3 Threshold Counting Method

To this point the method of threshold counting has proved to be the most advantageous. It offers the advantage of not penalizing the metric for small differences, on the order of the noise that may be present, and counting all values above this threshold equally. As in the two methods above, either the absolute value or squared difference method could be used, but due to its simplicity the absolute value method has been chosen. The procedure utilized produces an estimate that minimizes the expected value of the cost. The expected value of the cost is defined as the risk. That is,

$$R = E[C(I(D),I)] \tag{3.28}$$

3.3 DETERMINING INTEGER DISPLACEMENT

The method outlined above requires a good estimate of the integer displacement bounds. Several different methods have been tried for the purpose of establishing the displacement and a few of the various functional metrics, or cost functions, that can be used will be discussed below.

In each of the metrics described below, the previous frame search area is defined over a -p to p neighborhood for both spatial directions. The metric differencing is performed on a pixel by pixel basis for every pixel in the current block. A new value of the metric is generated every time the current block is shifted with respect to the previous frame block. The current block of size NxN is compared to all contiguous NxN subsets of the previous frame local neighborhood of size (2p+N) by (2p+N) as is shown in figure 3.4. Note that the metric generated will be a (2p+1) by (2p+1) matrix, where each entry's location is a function of displacement.

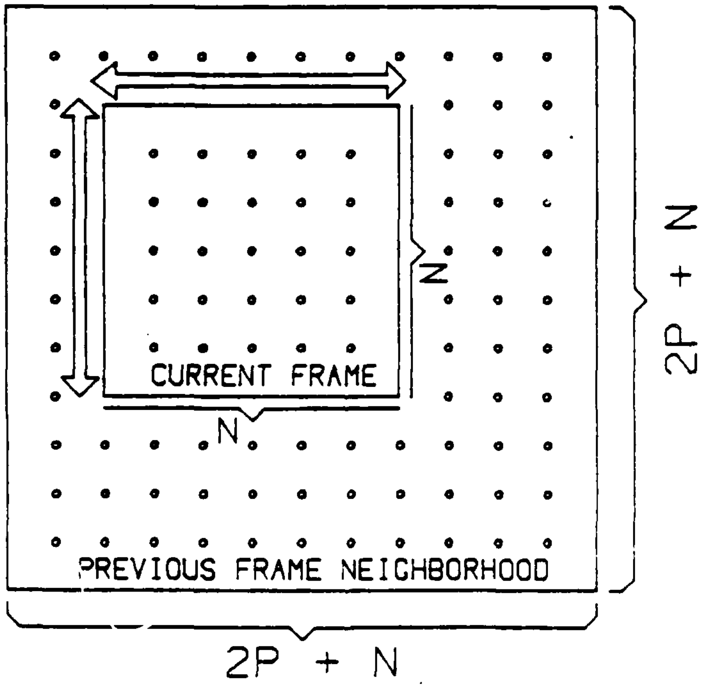


FIGURE 3.4 Pixel Neighborhood for Metric Determination

The problem of finding the noninteger portion of the displacement has been replaced by a two step process that is able to perform a similar function. First, for the one dimensional example, the actual displacement must be bounded above and below by a high and low integer estimate. Secondly the two values for the delta functions must be generated via the solution of the regression equation given below.

$$f(nT) = \sum_{i=0}^{1} a(i)\ g(nT - L + i) + e(nT). \tag{3.22}$$

Where L is the lower integer bound of the displacement estimate. Given a sufficiently large number of values for f(nT) and g(nT) the values for a(0) and a(1) can be found through least squares estimation. The same procedure can be extended to the two-dimensional process, where four on-sample delta functions replace the single off-sample function. As before note

$$I(i,j,t) = \delta(x,y)*I(i,j,t-\tau). \tag{3.23}$$

where x and y denote the location of the delta function which need not fall on the two-dimensional sampling grid. A time-modified autoregressive model taking into account that the prediction is made over the time boundary is then given by

$$I(i,j,t) = \sum_{m=-p}^{p} \sum_{n=-p}^{p} a(m,n)I(i-m,j-n,t-\tau) + e(i,j,t). \tag{3.24}$$

If, as before, the best integer estimate of the displacement bounds can be found, a displacement-updated, time-modified autoregressive model can be given as

$$I(i,j,t) = \sum_{m=0}^{1} \sum_{n=0}^{1} a(m,n)I(i-K+m,j-L+n,t-\tau) + e(i,j,t). \tag{3.25}$$

Here K and L designate the lower bounds of the integer displacement estimate. Here, as in the one dimensional case, the regression coefficients are able to perform the same function as the noninteger portion of the displacement vector.

For example, if the true value of the delta function occurs at $t = 1.35T$, it can be replaced by two different delta functions. One positioned at $t = 1T$, and the other at $t = 2T$. The relative value of each delta function is obtained from the following formulas, where $\delta_l(t)$ is a lower bound of the shift function and $\delta_h(t)$ is an upper bound of the true value of the shift.

$$\delta_l(t) = 1 - e \tag{3.20}$$

$$\delta_h(t) = e \tag{3.21}$$

In the example given above the value of the function at t=1T, labelled $\delta_l(t)$, would be .65, while the value at t=2T, labelled $\delta_h(t)$, would be .35 and hence is able to perform the function of linear interpolation as if it were a noninteger shift. Figure 3.3(b) shows the placement of the two off sample delta functions for the noninteger shift in 3.3(a).

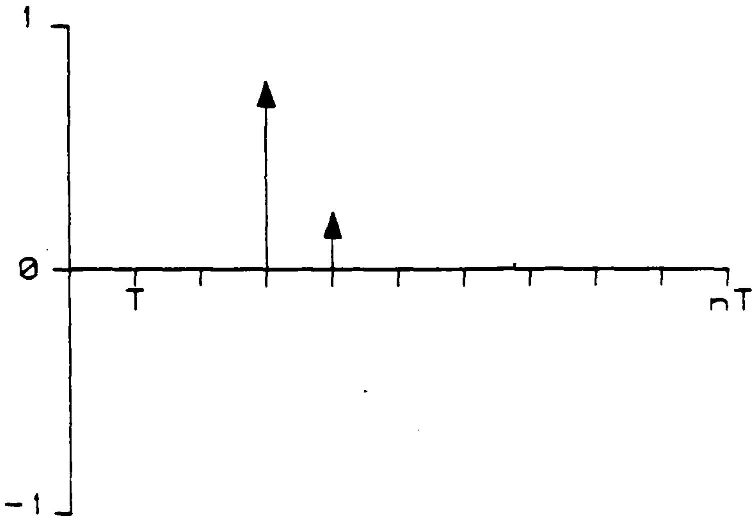


FIGURE 3.3(b) Delta Function for Noninteger Shift

The problem at hand uses this shifting and interpolation ability of multiple delta functions to bypass the need for sub-pixel determination of the shift vector. Assuming that a method can be found to determine the nearest integer displacement, with the use of regression analysis there is no need to find the noninteger portion of the displacement. Instead, only the magnitudes and locations of the delta functions are required. How the integer displacement is determined is given in the next section.

the use of linear interpolation of the two adjacent points. The form of the linear interpolation is given below where $f_b(n'T)$ designates the function value at the lower bound of the interpolation, $f_u((n'+1)T)$ designates the function value of the upper bound, and $f_m(n'T)$ is the function value at the linear interpolated point between these bounds. The n' is used here to denote a fixed value for n.

$$f_m(n'T+e) = f_b(n'T) + e(f_u((n'+1)T) - f_b(n'T)) \qquad (3.16)$$

In the above equation e is the distance from n'T to the desired interpolation point, or if you wish, the noninteger portion of the displacement. Carrying this idea further, an interpolated function can be generated by letting n' vary over the appropriate limits to include all values of the sampled function. Rewriting (3.16) in function form yields,

$$f_m(nT+e) = (1-e)f_b(nT) + ef_u((n+1)T). \qquad (3.17)$$

Note that in the two equations above, the shift or interpolation is limited to the range of 0 to 1. That is, e is bounded to the interval 0 to 1. From equations (3.16) and (3.17), it can be seen that the interpolated portion of the function is a linear combination of the points on either side. It is from this equation that the idea of multiple delta functions performing interpolation can be found. The first term on the right side of equation (3.17) can be modified by a delta function at the origin with value 1-e. The second part of (3.17) is modified by another delta function but at location t=T and of value e. Rewriting (3.17)in a convolutional form yields,

$$f_m(nT+e) = (1-e)f(nT)*\delta(t) + ef(nT)*\delta(t-T). \qquad (3.18)$$

Hence, the single off-sample delta function located at t=e has been replaced by two on-sample delta functions. This can be further extended to perform for any value of interpolation. That is, the displacement does not have to be restricted to the 0-1 range. When this is the case, e remains the noninteger portion of the displacement and both delta functions are then shifted by the remaining integer amount of the interpolation or shift. With S being the integer amount of shift, (3.18) can be written as follows.

$$f_m(nT+e) = (1-e)f(nT)*\delta(t-ST) + ef(nT)*\delta(t-(S-1)T) \qquad (3.19)$$

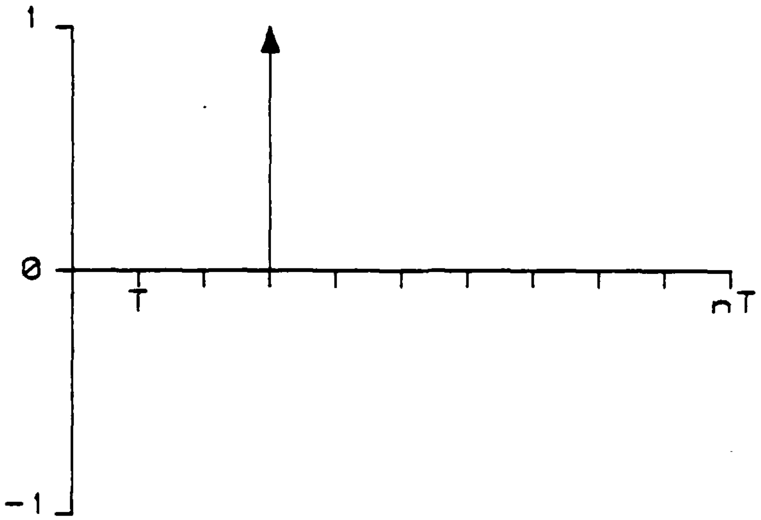


FIGURE 3.2(b) Delta Function for Integer Shift

function can be obtained by a linear combination of delta functions bordering the true time or phase shift. Using the method of linear interpolation, the single off-sample delta function can be replaced by two on-sample delta functions. The magnitude of each delta function is based on the location of the actual off-sample delta function.

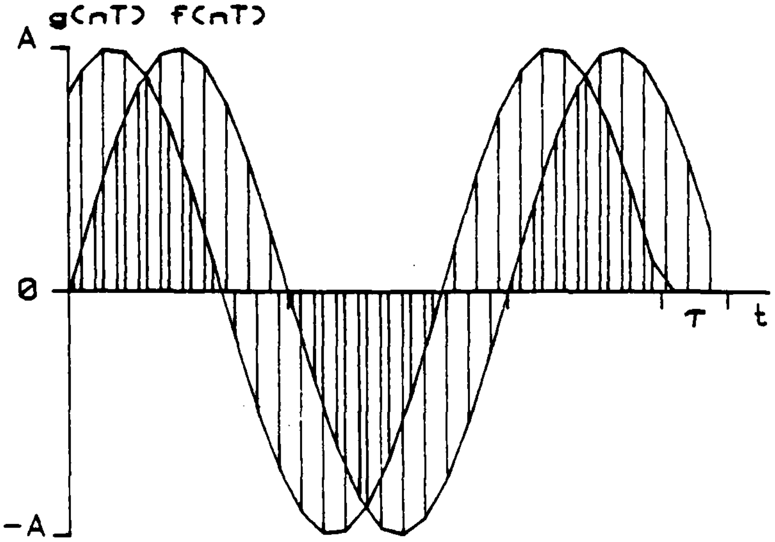


FIGURE 3.3(a) Sampled Function Noninteger Shift

Before the shift can be determined it is necessary to explain the mechanics of this linear interpolation process. It is assumed that any value of the function between sample points can be determined through

The graphical representation of this case is given in figure 3.2(a) and hence f(nT) and g(nT) for this specific example are given below.

$$f(nT) = A\cos(\tau nT + \theta) \tag{3.13}$$

$$g(nT) = A\cos(\omega(nT - \tau) + \theta) \tag{3.14}$$

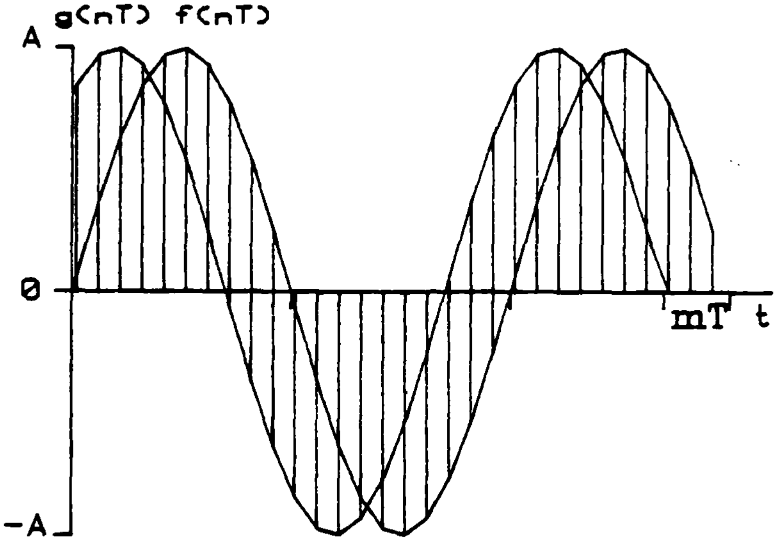


FIGURE 3.2(a) Sampled Function Integer Shift

As long as τ is an integer multiple of the sampling period T, as shown in figure 3.2(b), the same holds true for the sampled case as for the continuous case and hence,

$$f(nT) = g(nT)*\delta(t - m\tau) \tag{3.15}$$

where $m\tau$ in an integer multiple of T.

Figure 3.3(a) graphically depicts the usual case in that the shift is a noninteger multiple of the sampling period. The problem here is that $\delta(t-\tau)$ may not fall on a sampled time interval. Example $\tau = 1.5T$ or 1.5 times the sampling interval and hence would lie halfway between the two sampled time intervals. The method presented here attacks this problem of noninteger shift with the help of linear interpolation. Assuming that all points of the sampled time function between samples can be determined sufficiently close by linear interpolation of the sample points on each side, then a sufficiently good estimate of the shifted

$$g(t) = A\cos(\omega(t - \tau) + \Theta) \tag{3.6}$$

Equation (3.4) for this specific example then becomes,

$$f(t) = A\cos(\omega(t - \tau) + \Theta)*\delta(t-\tau) \tag{3.7}$$

Where $\delta(t-\tau)$ is shown in figure 3.1(b).

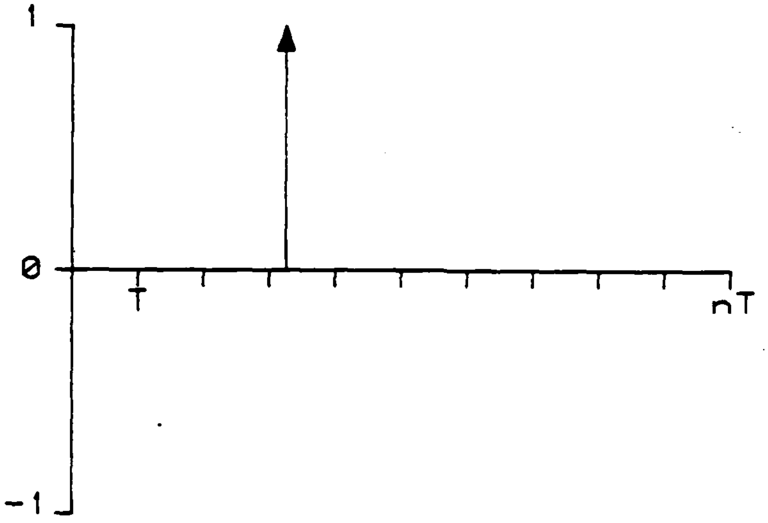


FIGURE 3.1(b) Delta Function for Continuous Shift

Using the Fourier transform to perform time convolution yields

$$f(t) = F^{-1}[F[A\cos(\omega(t - \tau) + \Theta)]\ F[\delta(t - \tau)]] \tag{3.8}$$

$$f(t) = A\cos(\omega t + \Theta) \tag{3.9}$$

where F designates the forward Fourier transform and f^{-1} the inverse transform. This is the case as was given in equation (3.5).

Getting closer to the problem at hand, the sampled versions of f(t) and g(t), namely f(nT) and g(nT), obtained from the sampling function,

$$f(nT) = f(t)\sum\delta(t-nT) \tag{3.10}$$

can be written as follows.

$$g(nT) = f(nT-\tau) \tag{3.11}$$

or

$$f(nT) = g(nT)*\delta(nT). \tag{3.12}$$

examining a single dimensional continuous time function example with the help of linear systems theory. The two-dimensional, noiseless, no background case becomes, in a single dimension, the problem of determining the phase shift between two identical, but time shifted, time functions. For example if

$$g(t) = f(t-\tau) \tag{3.3}$$

the two functions are said to be identical, with the exception of the time shift. The problem then of finding the time shift amounts to finding the value for τ. Using the concept of convolution, g(t) and $f(t-\tau)$ can be related further by

$$f(t) = g(t)*\delta(t-\tau) \tag{3.4}$$

where the function * is defined to be convolution and $\delta(t-\tau)$ is a delta function located at $t=\tau$. The problem remains to determine the time displacement from the origin to the location of the delta function. Figure 3.1(a) graphically presents a possible representation of these two continuous time functions shifted in time by τ. This displacement can be determined through the use of correlation or a matched filter *such that the output is the phase shift between the two signals.*

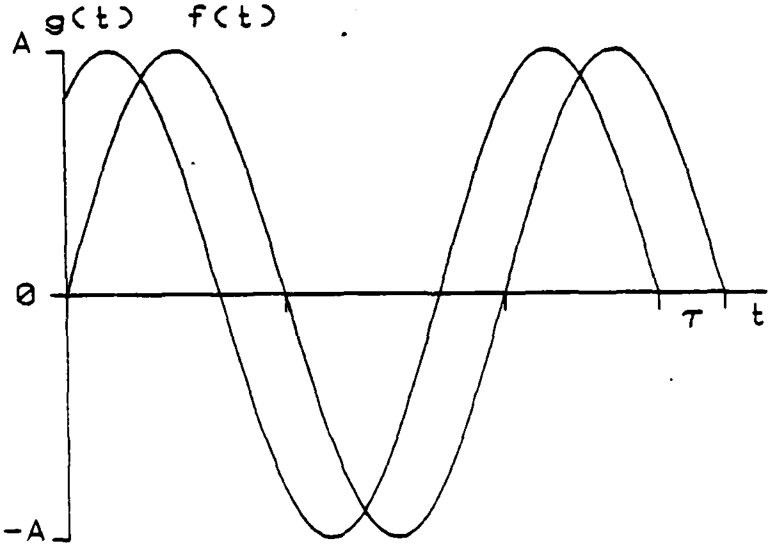


FIGURE 3.1(a) Continuous Function Shift Example

From the example in figure 3.1(a) f(t) and g(t) can be written as below.

$$f(t) = A\cos(\omega t + \theta) \tag{3.5}$$

3.2 ALGORITHM DESCRIPTION

In the methods previously discussed for motion compensated image coding, [91], [112], [113], and [116], the current image pixel is modelled as a pixel from the previous frame displaced by some displacement vector D. This can be shown as in (3.1).

$$I(X_k, t) = I(X_k - D,\ t - \tau) \tag{3.1}$$

Where $I(X_k, t)$ is the pixel intensity at location X_k and τ is the time delay between adjacent frames. This functions adequately provided the displacement is of integer order. That is; no partial pixel displacements are allowed. If sub-pixel displacements do exist, this necessitates the finding of a D vector that is also of sub-pixel order. A somewhat more complicated approach, but offering the advantage of not having to determine a displacement vector, can be found. The current frame pixel intensity is defined to be a linear combination of pixel intensities from the previous frame.

$$I(X_k, t) = \sum_{m=1}^{M} \sum_{n=1}^{N} a(m,n) I(X_k(m,n), t-\tau) + e(X_k, t) \tag{3.2}$$

The disadvantage here is that although no motion vector is required, a set of predictor coefficients a(m,n) is required. If the system is to allow for a p pixel shift in any direction, the size of this 'a' matrix would be at least (2p+1) by (2p+1). Hence, the amount of data compression that can be achieved is greatly diminished as p gets large. Note that this prediction matrix must be transmitted every time the displacement changes between blocks.

The ideal situation would be to take advantage of the linear combination method, so that the accuracy of the motion vector can be kept to integer displacements, and yet exploit the data compression that the displacement vector approach offers. Going back to (3.2) it can be seen that the solution of the problem involves the determination of the prediction matrix a(m,n). This results in a two-dimensional regression problem. Noting what the physical implications are in relation to the regression problem, it can be seen that the regression coefficients would be used to perform translation and hence the model can be simplified. A better understanding of this concept can be obtained by

3.1 UNDERLYING ASSUMPTIONS AND REQUIREMENTS

Before getting into the actual algorithm, there are a set of assumptions and requirements that need to be stated.

1. Each image in the sequence is broken up into a set of square blocks. The motion for that block is assumed to be constant over the entire block, regardless of the chosen blocksize.
2. The motion as modelled by the algorithm is pure translation, that is only motion that is translation or can be modelled by translation between frames is actually modelled. Nontranslation type motion and problems of occlusion will be handled by the quantization of the residual and not entirely by the displacement vector. Although in some cases, the prediction coefficients can somewhat compensate for nonideal motion.
3. It is further assumed that the maximum displacement between frames is known. The system is no more complex for large allowable displacements, but the number of calculations increases rapidly as the maximum allowable displacement increases. Hence, the maximum displacement estimate should be kept as small as possible to improve performance. For a majority of image work a value of five or six pixels is appropriate.
4. It is known that the human visual system will accept a higher degree of degradation for scenes undergoing translation than it will for static scenes. For this reason, the blocks that are undergoing translation are allowed a lower signal to noise ratio than those which do not move. That is, the moving blocks can have a higher degree of degradation than the stationary ones.

With this set of assumptions in hand, the goals of the algorithm can be stated: the overall system should be able to obtain a fairly high degree of data compression, but not at the expense of overall picture quality. The system should be able to achieve a very low data rate for image sequences with little or no motion. For the blocks in the image that are undergoing translation the method should be able to determine the motion vector to sub-pixel accuracy and code the resulting output at a sufficiently low data rate.

Chapter III

PREDICTION COEFFICIENT ENERGY CONCENTRATION

In the last chapter some of the previous attempts at motion estimation, and in particular motion compensated image coding, were presented, setting the groundwork material for this chapter. The method of 'Prediction Coefficient Energy Concentration' differs from the previous methods not in terms of optimal output, but only in the way in which this goal is achieved.

As the name 'Motion Compensated Image Coding' implies, motion or movement estimation is used to decrease the data rate required in the transmission of time sequential image data. The use of this motion or movement requires that a good estimate for the displacement be made available to the image coder. It is the finding of this displacement vector that is new and original in this work. Previous work only required the displacement to be found to the nearest integer multiple of the pixel spacing. This may suffice for some applications, but where the human observer is the final link in the coding system, these integer only displacements tend to be somewhat annoying. For this reason, and others concerned with actual displacement measurement, the noninteger portion of the displacement is needed. The methods of pel recursive and coefficient recursive displacement estimation use an iterative recursive minimization procedure to converge to a displacement that is of noninteger order.

In this chapter a new method for finding the displacement vector for use in a motion compensated image coding system is introduced and investigated. The method of prediction coefficient energy concentration differs from the earlier recursive methods in that a solution for the displacement is explicitly defined and does not have to iterate to a solution. Further, it differs by the fact that no actual noninteger portion of the displacement actually has to be found. Instead this is replaced by a coefficient prediction process that obtains the same results, or in many cases better, with no increase in the algorithm complexity.

metric representing each corner of each quadrant where the center value, being common to all quadrants, may be neglected as is shown in figure 3.5. Each quadrant then is represented by a sum of three metric values and the minimum quadrant is defined to be the one with the minimum sum. This in essence determines the integer bounds or integer displacement cell for the noninteger portion of the displacement. In other words, the true value of the displacement is not known, but it is assumed to fall within the bounds of the displacement cell. Hence the locations of the four delta functions are defined to be at each corner of the displacement cell. This yields the locations of the delta functions, but not their magnitudes. The method for determining their magnitudes is given in the following section.

At this point, it is informative to look at a two-dimensional example for the displacement of a single pixel. Figure 3.6(a) presents a reference pixel from the current block. Figure 3.6(b) shows the displacement associated with this pixel as well as the integer estimates for the X and Y displacements. The four pixels that define the box are used in a linear combination to estimate the reference pixel.

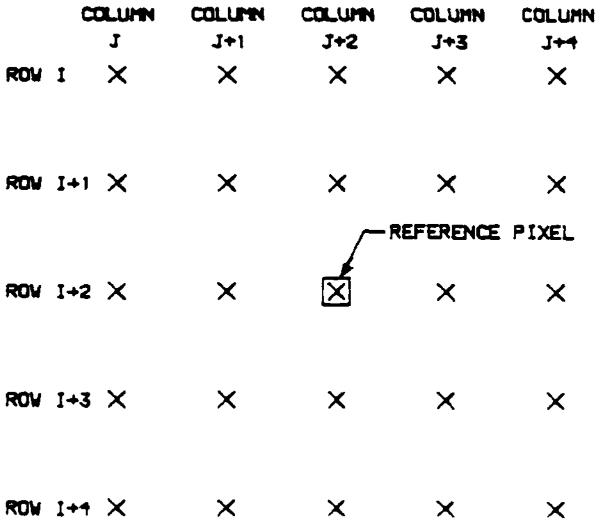


FIGURE 3.6(a) Current Frame Pixel Location

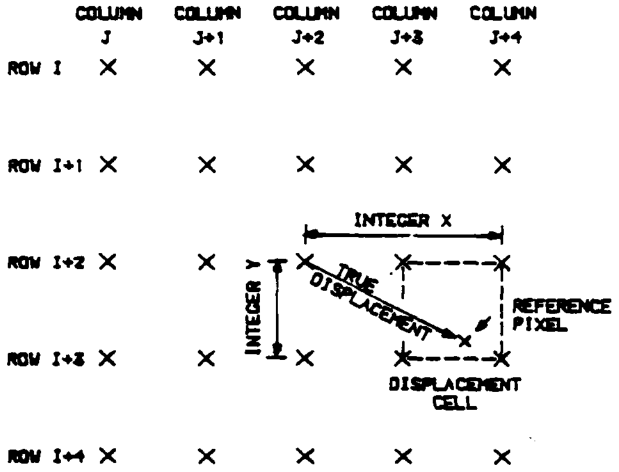


FIGURE 3.6(b) Displacement Measurements for Reference Pixel

3.5 PREDICTOR COEFFICIENT GENERATION

Having bounded the displacement estimation in both directions, the time-modified, translation-updated regression model is given as before.

$$I(i,j,t) = \sum_{m=0}^{1} \sum_{n=0}^{1} a(m,n)I(i-K+m,j-L+n,t-\tau) + e(i,j) \qquad (3.35)$$

Where, as before, K and L designate the lower bounds on the displacement estimation. As was noted earlier, the assumption is made that the displacement remains constant over the block and hence the regression coefficients will also remain constant over that block.

3.5.1 The Regression Problem

The solution for a single dimensional regression process is straight forward and will not be addressed here. Instead, only its relationship to the problem at hand will be discussed.

The matrix equation normally used for single dimensional regression can be written as follows

$$Y = XB + e \qquad (3.36)$$

where Y is a dependent variable vector of size n. X is the augmented independent variable matrix of size n by p, where p is the number of regression coefficients. B is then the parameter vector or vector containing the predictor coefficients and is also of size p. Finally e is the error vector or residual. The matrix normal equation is given by

$$X^TXb = X^TY. \tag{3.37}$$

The least squares estimate for B is b and can be obtained from

$$b = (X^TX)^{-1}X^TY. \tag{3.38}$$

A further restriction is placed on the current regression problem in that it is an autoregressive series, or actually an autoregressive spatial-time series, and current values are predicted from past time values from the same general spatial location. The autoregressive equation can be written as

$$Y = ZB + e. \tag{3.39}$$

Z represents a shifted version of Y, in this case both spatially and temporally. In the particular case at hand, the derivation has to be carried one step further, because it must take into account that the data is two-dimensional and the prediction is made over a time boundary. Note that the regression equation given in (3.35) is not in matrix form. Some data manipulation is required if standard methods for regression analysis are to be used.

3.5.2 Method for Data Manipulation

First the *coefficient* matrix $a(i,j)$ needs to be placed in a vector fitting the description of the B vector in (3.36), therefore

$$B = \begin{bmatrix} a(0) \\ a(0,0) \\ a(0,1) \\ a(1,0) \\ a(1,1) \end{bmatrix} \tag{3.40}$$

where it is augmented by a(0), the intercept or bias term. Next, the Z matrix needs to be set up to perform the shifting that is accomplished by the double summation in (3.35). In the equation for the Z matrix that follows, the $t-\tau$ factor and lower integer bound terms are neglected for simplicity.

$$Z = \begin{bmatrix} 1 & I(i,j) & I(i,j+1) & I(i+1,j) & I(i+1,j+1) \\ 1 & I(i,j+1) & I(i,j+2) & I(i+1,j+1) & I(i+1,j+2) \\ & \vdots & \vdots & \vdots & \vdots \\ 1 & I(i,j+N) & I(i,j+N+1) & I(i+1,j+N) & I(i+1,j+N+1) \\ 1 & I(i+1,j) & I(i+1,j+1) & I(i+2,j) & I(i+2,j+1) \\ & \vdots & \vdots & \vdots & \vdots \\ 1 & I(i+N,j+N) & I(i+N,j+N+1) & I(i+N+1,j+N) & I(i+N+1,j+N+1) \end{bmatrix} \qquad (3.41)$$

Figure 3.7 is provided to show how the Z matrix is generated from the two dimensional image intensity values. Note, the first column of Z is augmented with 1's to correspond to the intercept or bias term of the B vector.

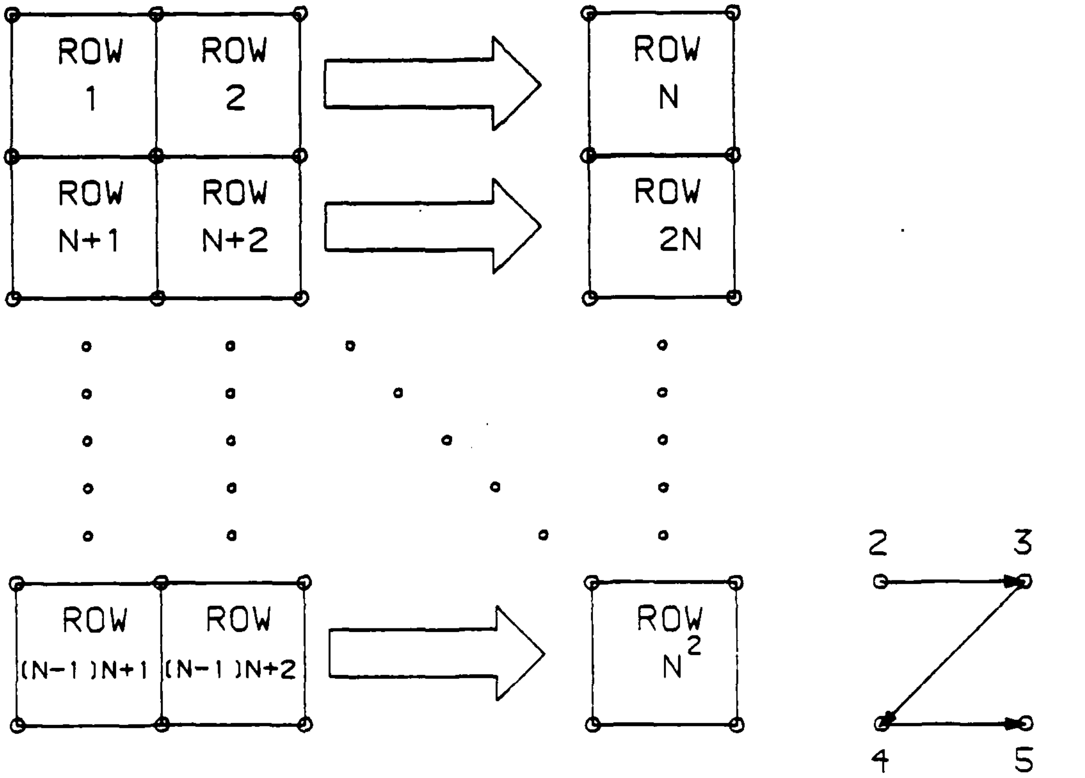


FIGURE 3.7(a) Z Matrix Scan Diagram (b) Single Row Scan

Filling in the remaining terms of the Z matrix one row at a time starting from the top and working down can be accomplished with the help of the scan diagrams given in figure 3.7. Figure 3.7(a) represents the pixel groupings by row for the intensities from the previous frame. Each row of Z is represented by a box in figure 3.7(a). An expanded version of one of these boxes is shown in figure 3.7(b) where each corner of the box designates a column in the Z matrix. Hence, for each row in Z there is a corresponding group of four pixels whose intensity values are to be placed into the column associated with the number given for each corner, as is shown in figure 3.7(b). Note that each pixel intensity may be used up to four times in the Z matrix. Finally, row scan the current frame block placing each value into the column vector Y as shown.

$$Y = \begin{bmatrix} I(i,j,t) \\ I(i,j+1,t) \\ \cdot \\ \cdot \\ I(i,N,t) \\ I(i+1,j,t) \\ \cdot \\ \cdot \\ I(i+N,j+N,t) \end{bmatrix} \qquad (3.42)$$

The scanning diagram for the current frame block is given below in figure 3.8. The residual vector e is defined identically to that of the Y vector.

With each of the variables Y, Z, B, and e defined, the current problem reverts to that of a normal one dimensional autoregression problem involving five coefficients and hence can be solved as such. The least squares estimate for B is b and is given by

$$b = (Z^T Z)^{-1} Z^T Y. \qquad (3.43)$$

As with any system that requires a matrix inverse, it is possible that the system may become ill conditioned and the inverse may not exist. In the current system, this is remedied by using only the estimate for the integer portion of the displacement. When this occurs, the b vector is set to an identity transfer function, that is,

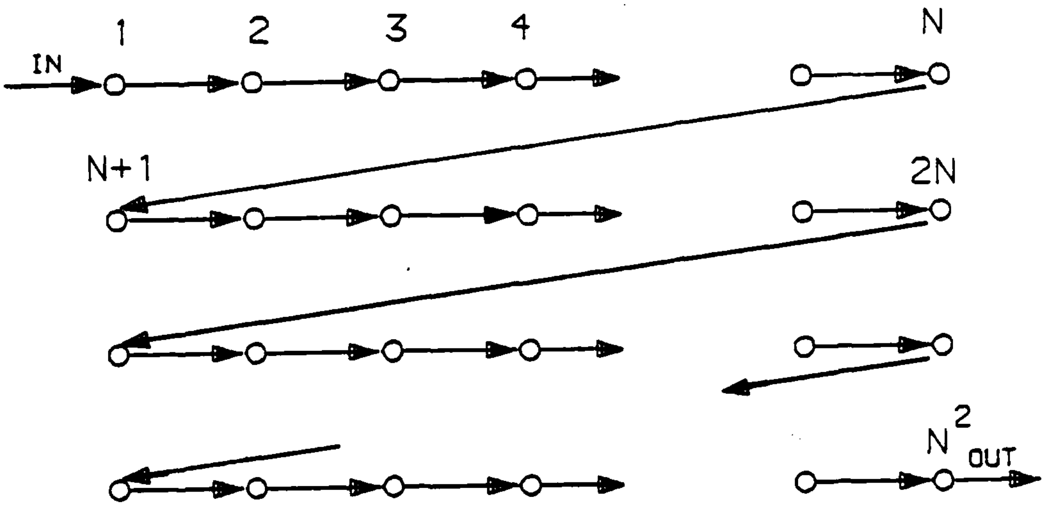


FIGURE 3.8 Y Vector Scan Diagram

$$b^T = [0,1,0,0,0] \qquad (3.44)$$

and amounts to using only the information from the integer displacement estimate in the prediction process.

3.6 BLOCK PREDICTION AND RESIDUAL GENERATION

After the completion of the identification portion of the system, it is necessary to generate the data required for transmission. The data required by the receiver is broken up into two separate parts. The first part is the receiver control block, which contains the information generated by the identification portion of the system. The remaining part is what is termed the residual data block. It contains the information required by the receiver to update or correct the block estimate when based only upon the control block information. The receiver, as well as the receiver portion of the transmitter, takes the control block information and produces an estimate of the current block using a similarly spatially located block from the previous reconstructed frame.

$$e(i,j,t) = I(i,j,t) - \sum_{m=0}^{1} \sum_{n=0}^{1} a(m,n)I(i-K+m,j-L+n,t-\tau) - a(0) \qquad (3.45)$$

This residual term is then quantized for transmission.

In order to fully exploit the between frame redundancy, checks are placed in the system to flag the types of changes that occur. The first check determines if any motion has occurred in the current block. If no motion is found, the process simply goes to the next block with no required data transmission. If motion has occurred, it is tested to determine if an integer estimate is a sufficiently good estimate. If it is deemed an integer only displacement, the block address and displacement are transmitted to the receiver. When integer displacement is not accurate enough, the prediction coefficients must be calculated. At this point the transmitter must send the block address, integer displacement, quadrant, predictor coefficients and possibly a quantized version of the prediction error. A final check is performed to determine if the error is still too large. If it is, the primary blocksize is cut in half and the process is retried for each of the four sub-blocks.

3.7 CODING

As noted in the previous section, there are two types of information that must be transmitted to the receiver. The first is the control block information, which contains the information required to make the motion compensated estimate and the second is the residual data block. The residual data block is the quantized error of the actual prediction, along with the coding information needed by the quantizer. Each of these will now be discussed in detail below.

3.7.1 Control Block Information

The control block information data sequence contains the set up variables needed by the receiver in order to start the prediction process. The data sequence is of variable length, depending on the mode of operation, of which there are four. As noted earlier, if no change has occurred for a block, then the receiver requires no update.

The four different modes of operation can be separated into two groups of two categories. The two groups are integer and noninteger displacements, while the two categories are the different blocksizes of 8 by 8 and 16 by 16. Table 3.1 shows the bit requirements for each mode of operation and the bit breakdown. The assignment of the first 10 bits remain the same for all four modes. These first 10 bits consist of a bit for integer/noninteger mode, a blocksize bit for blocksize determination and 4 bits each for the integer X and Y displacement. This will allow for a shift in either direction of from minus seven to plus seven pixels. The remainder of the bit structure differs in accordance with the blocksize bit and integer displacement bit. These remaining bits contain the block address for the current block and, if the noninteger bit is set, the quadrant number. Four of the predictor coefficients are coded into this section, while the fifth, since it is used by the residual quantizer, is coded in the residual block section. The number of bits used is based on an image size of 256 pixels square or smaller. If a larger image is used, then appropriate changes will require the enlargement of the block address.

This control block is essentially the overhead required for the prediction process and in itself produces a very low data rate. This overhead is zero for the motionless blocks and can range from about 0.07 up to 0.98 bits per pixel for blocks that contain motion. Hence, the predictor overhead will normally contribute only a small portion of the total data rate requirements. The majority of the bandwidth required is a result of the prediction error and hence is required by the residual data block.

TABLE 1

	INTEGER		NON-INTEGER	
BIT	8x8	16x16	8x8	16x16
1	0-Integer only	0-Integer only	1-Non-Integer	1-Non-Integer
2	0-Blocksize = 8	1-Blocksize = 16	0-Blocksize = 8	1-Blocksize = 16
3	X	X	X	X
4	X X Direction	X X Direction	X X Direction	X X Direction
5	X Displacement	X Displacement	X Displacement	X Displacement
6	X	X	X	X
7	Y	Y	Y	Y
8	Y Y Direction	Y Y Direction	Y Y Direction	Y Y Direction
9	Y Displacement	Y Displacement	Y Displacement	Y Displacement
10	Y	Y	Y	Y
11	X	X	X	X
12	X X Block	X X Block	X X Block	X X Block
13	X Location	X Location	X Location	X Location
14	X	X	X	X
15	X	Y	X	Y
16	Y	Y Y Block	Y	Y Y Block
17	Y Y Block	Y Location	Y Y Block	Y Location
18	Y Location	Y	Y Location	Y
19	Y		Y	C
20	Y		Y	C Prediction
21			C	C Coefficients
22			C Prediction	C
•			•	•
62			C	C
63			R Residual Required if=1	

3.7.2 Residual Data Block

When the cumulative error between the predicted block and actual block exceeds a set threshold, a residual block must also be transmitted. This data is used to correct some of the errors that occur in the prediction process. The block consists of two parts. The first contains the mean and variance of the residual signal and the number of quantizer levels. The remaining data is the quantized residual data. Fixed rate quantizers are inappropriate because the residual signal is non-stationary. A variable rate quantizer, based on the mean and variance of the residual signal and the predictor gain, performed quite well in spite of the non-stationary signal. For very small variance errors no residual was needed. For larger errors, up to 7 bits were available per pixel.

Due to the very large variation in the quality of the predicted signal, an adaptive-variable-length quantizer is required in order to maintain a minimal data rate for a specified overall picture quality. Here, as in much of the other work in image processing, there is no good clear cut numerical indication of picture quality. For the work reported here two statistical indicators were jointly used for internal judgement of picture quality. It is upon the basis of this judgement that the residual data rate is determined. The first statistical indicator is the error signal variance. It is expected that a low variance value means good image reproduction while a large error variance indicates a large prediction error and hence a large residual data rate requirement. The second statistical indicator is the prediction gain or predictor signal to noise ratio. While this does take into account the error variance from above, it also takes into account the signal variance and is defined by

$$\mathrm{GAIN} = 10\mathrm{Log}(\mathrm{var}(\mathrm{signal})/\mathrm{var}(\mathrm{error})). \qquad (3.46)$$

Using these two quality indicators as fidelity criteria, the adaptive quantizer determines the required number and placement of quantizer output levels in order to achieve a specified output signal to noise ratio. The residual variance is used as a simple yes/no indicator for the quantizer. That is, if the error variance exceeds the preset threshold, the quantizer will be used to transmit the residual. The predictor signal to noise ratio is then used by the quantizer, along with the minimum acceptable signal to noise ratio, to determine how much the predicted signal needs to be improved. This is equivalent to determining the quantizer level requirements. The number of levels is determined by finding the minimum number of levels needed to obtain the preset output signal to noise ratio, provided the error distribution is Gaussian. Even though the global error distribution may be gaussian, rarely will the local distribution be so. For this reason, rarely will the output signal to noise ratio match that which was preset. However, this will bound the actual data rate requirements.

The quantizer employed in the actual implementation is a multiple level and hence variable rate. The quantizer chooses the thresholds and output levels optimally based on a distortion measure defined to be the sum of the absolute values of the error. The choice of this distortion measure over the standard mean squared error results in a decrease of the granular noise in the areas of the image with small signal variance. Figures 3.9(a) and (b) are provided to show the differences in output error distributions for the different distortion measures. Each is assumed to have a Normal(0,1) input error distribution. The curves labelled 0,1,2,3, and 4 bits are the error output distributions after modification by the selected quantizer. If an infinite number of bits were used, the output error distribution would be a delta function centered about 0. Note that the mean square method tends to minimize the area under the tails of the distribution, while the absolute value method tends to concentrate more on the center portion of the distribution. As noted earlier, the number of quantizer levels required is based upon the error variance and the predictor signal to noise ratio.

When the error variance exceeds the set threshold, the difference between the requested gain and the actual predictor gain is in a sense a 'gain' that must be generated by the quantizer. Gain is a term not normally associated with quantizers, but in this case it is used to compare the output quality with a given quantizer to the quality that would be present if no quantizer were used. That is to say, how much is the signal improved by using a quantizer to send the error as compared to not sending the error at all.

This gain is determined as a function of the number of output levels. It is then a simple procedure to determine how many levels are required to meet gain requirements to obtain this reproduced image fidelity criterion. A plot of this gain as a function of output levels is given in figure 3.10. From this, for a required gain, the number of output levels can be determined. The similarity of this curve to the log rate distortion curve is apparent.

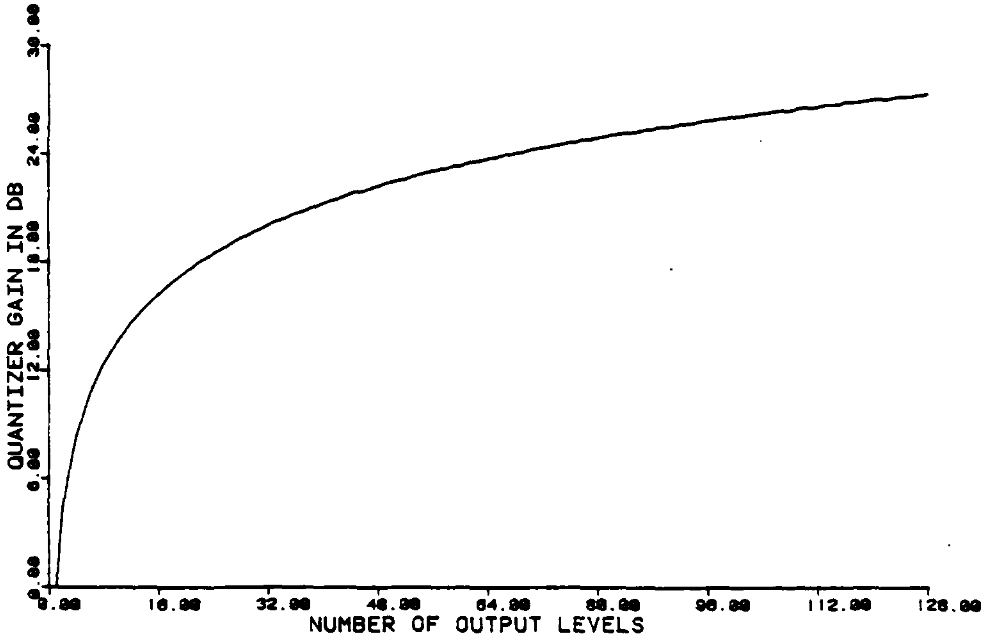


FIGURE 3.10 Quantizer Gain as a Function of Number of Output Levels

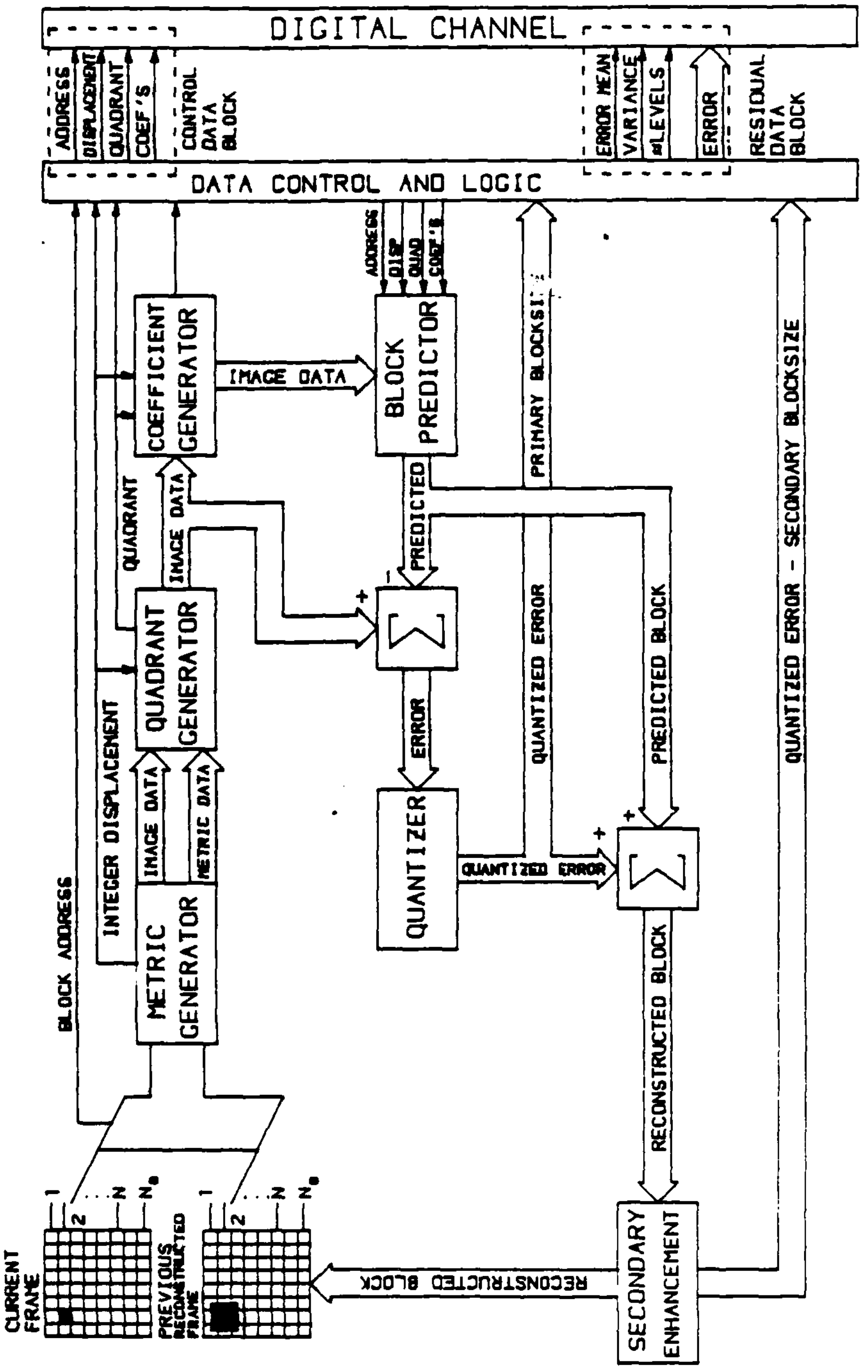
Chapter IV
EXPERIMENTAL OPERATION

The presentation of results obtained from most forms of image processing is a difficult one, but even more so when the data consists of time varying image sequences, as is the case in motion compensated image coding. Comparisons among published results are even more difficult because of the lack of standard presentation procedures and the lack of standard test image sequences. The performance of any motion compensated coding system is extremely scene or sequence dependent and hence, this dependence as a function of system and scene characteristics will be presented.

The image sequences used for this system performance analysis consist of 41 images of size 128 by 192 pixels quantized to 8 bits. The first sequence is termed 'SLOW PHONE' and consists of a woman talking on a phone while slowly rotating and moving the phone. The second sequence, termed 'FAST PHONE', is similar to the first but contains faster and more motion. Both sequences contain very complex motion with foreground-background interaction and are similar to those that would be encountered in a video-phone setting.

A block diagram showing each of the subsystems mentioned in the previous chapter and their relationship to one another is given in figure 4.1. Note that the block diagram presents each subsystem serially connected with no indication for mode of operation. The mode, recall there are four, is controlled by the data control and logic block. Note also that the system can be easily implemented in parallel in terms of block operations because one block prediction is not based on the previous block from the same frame, but only on blocks from the previous reconstructed frame.

As stated above, the presentation of the results for time varying imagery work is very difficult. When images from the sequence are viewed singularly, they appear 'cleaner' or less noisy than they would if viewed in the actual sequence. Also, some prediction errors that do appear in the single frames are not as noticeable when the frames are


DIGITAL CHANNEL
ADDRESS
DISPLACEMENT
QUADRANT
COEF'S
CONTROL DATA BLOCK
ERROR MEAN
VARIANCE
#LEVELS
ERROR
RESIDUAL DATA BLOCK
DATA CONTROL AND LOGIC
ADDRESS
DISP
QUAD
COEF'S
COEFFICIENT GENERATOR
IMAGE DATA
BLOCK PREDICTOR
PRIMARY BLOCKSIZE
QUANTIZED ERROR - SECONDARY BLOCKSIZE
QUADRANT
IMAGE DATA
PREDICTED
+
-
QUADRANT GENERATOR
INTEGER DISPLACEMENT
ERROR
QUANTIZED ERROR
PREDICTED BLOCK
IMAGE DATA
METRIC DATA
QUANTIZER
QUANTIZED ERROR
+
+
BLOCK ADDRESS
METRIC GENERATOR
RECONSTRUCTED BLOCK
1
2
N
N_B
1
2
N
N_B
CURRENT FRAME
PREVIOUS RECONSTRUCTED FRAME
RECONSTRUCTED BLOCK
SECONDARY ENHANCEMENT

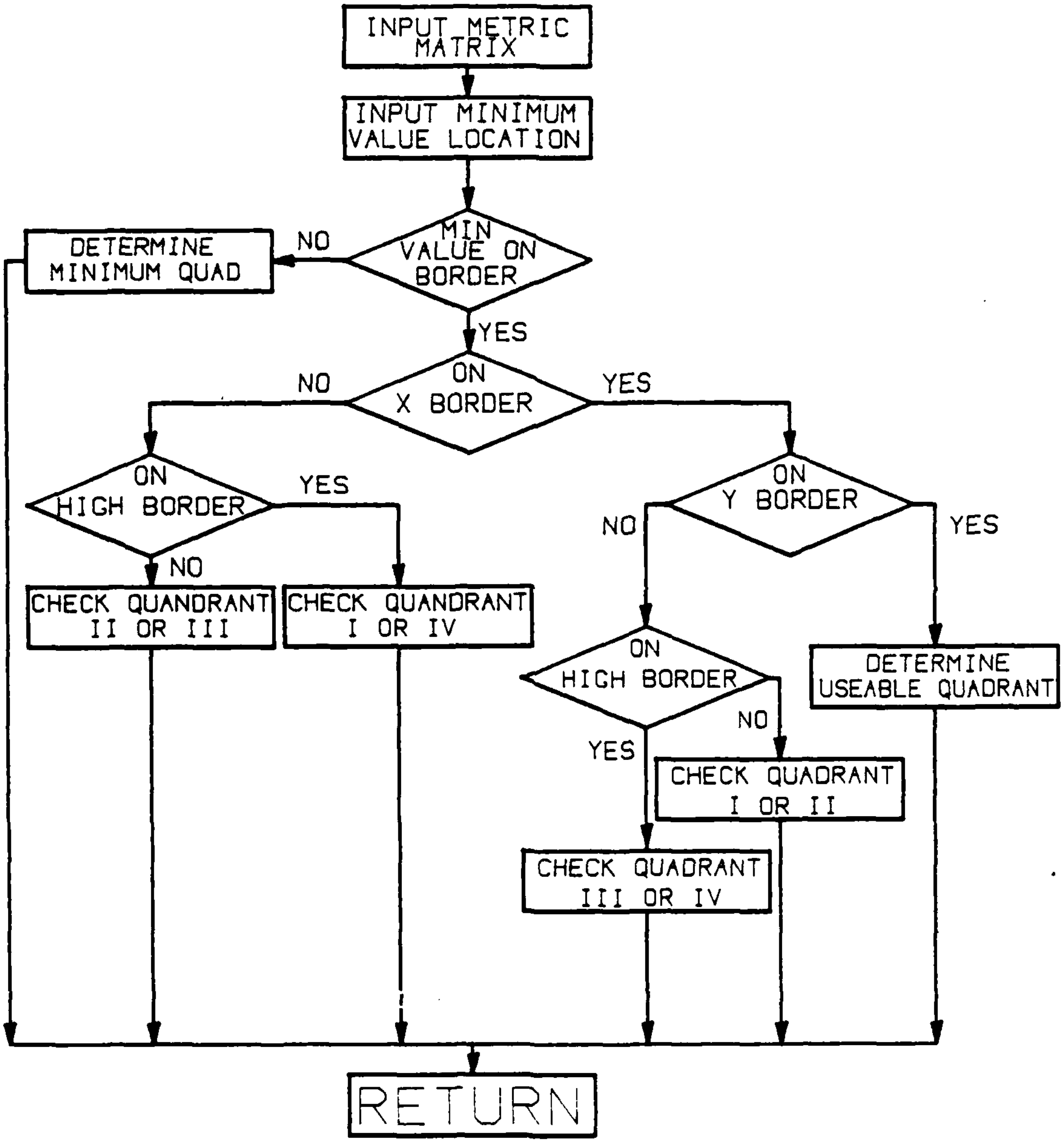


FIGURE 4.7 Quadrant Determination Flow Chart

4.4 MINIMUM QUADRANT GENERATION

Upon return from the metric subroutine, an estimate of the integer displacement is provided. The subroutine also supplies a variable that is equal to the value of the metric at its minimum location. This value can be used to determine how good of an estimate of the current block can be generated using the previous frame and the integer displacement estimate. For many blocks this estimate may prove to be sufficient. For others, it indicates that more information will be required in the image reconstruction. For the blocks that the estimate is sufficient, no further displacement estimation is required. For the others, further computations are required to improve the current frame estimate. This further computation may involve determining a better estimate for the displacement vector or, as in the method presented, solve for the regression parameters.

When an integer-only displacement estimate is not good enough, the next step is to calculate the best estimate for the quadrant in which the true displacement falls. This is accomplished through the use of the LOCATE subroutine as given in appendix B. The accompanying flowchart is provided in figure 4.7. Recall from chapter 3 that this quadrant is generated by summing the values of the metric that surround the best integer estimate. The quadrant is used to bound the displacement estimate to a single integer displacement cell. This subroutine locates the quadrant with the lowest sum of surrounding pixel values. It does not determine the values for the coefficients.

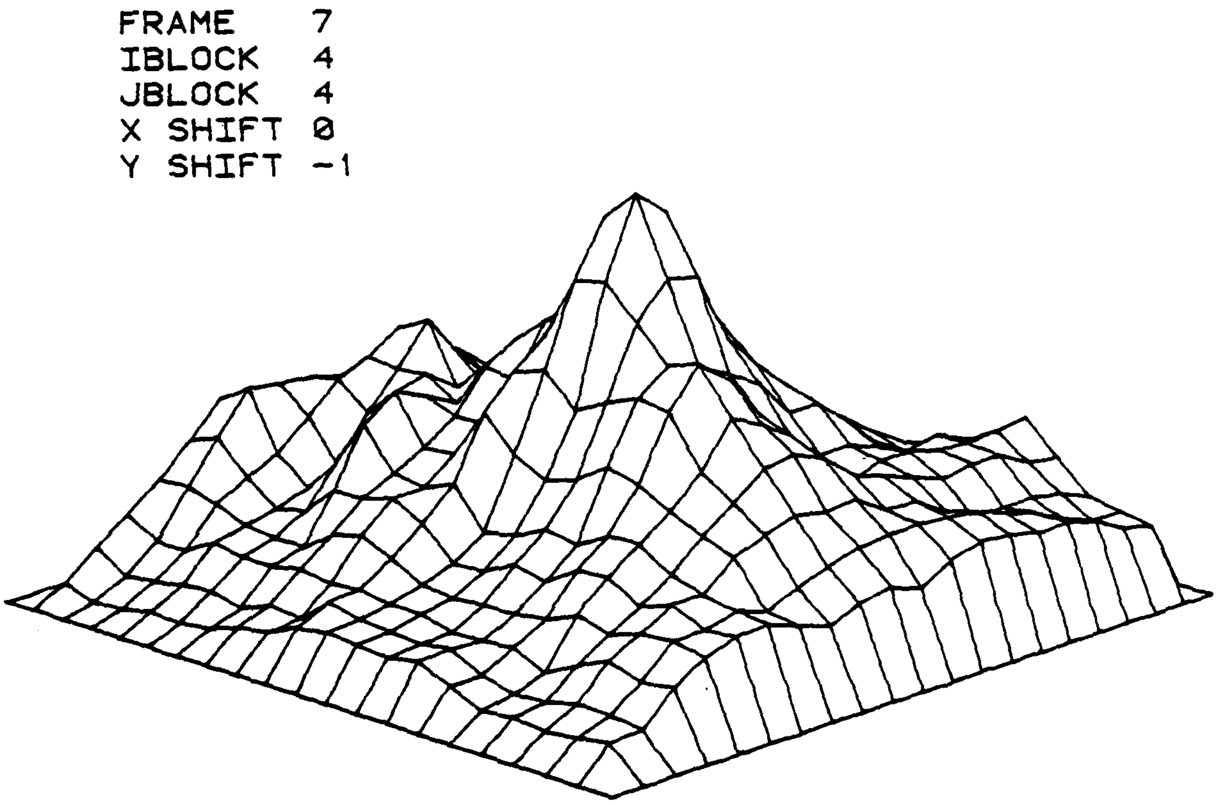


FIGURE.4.6(c) Plot of Actual Metric

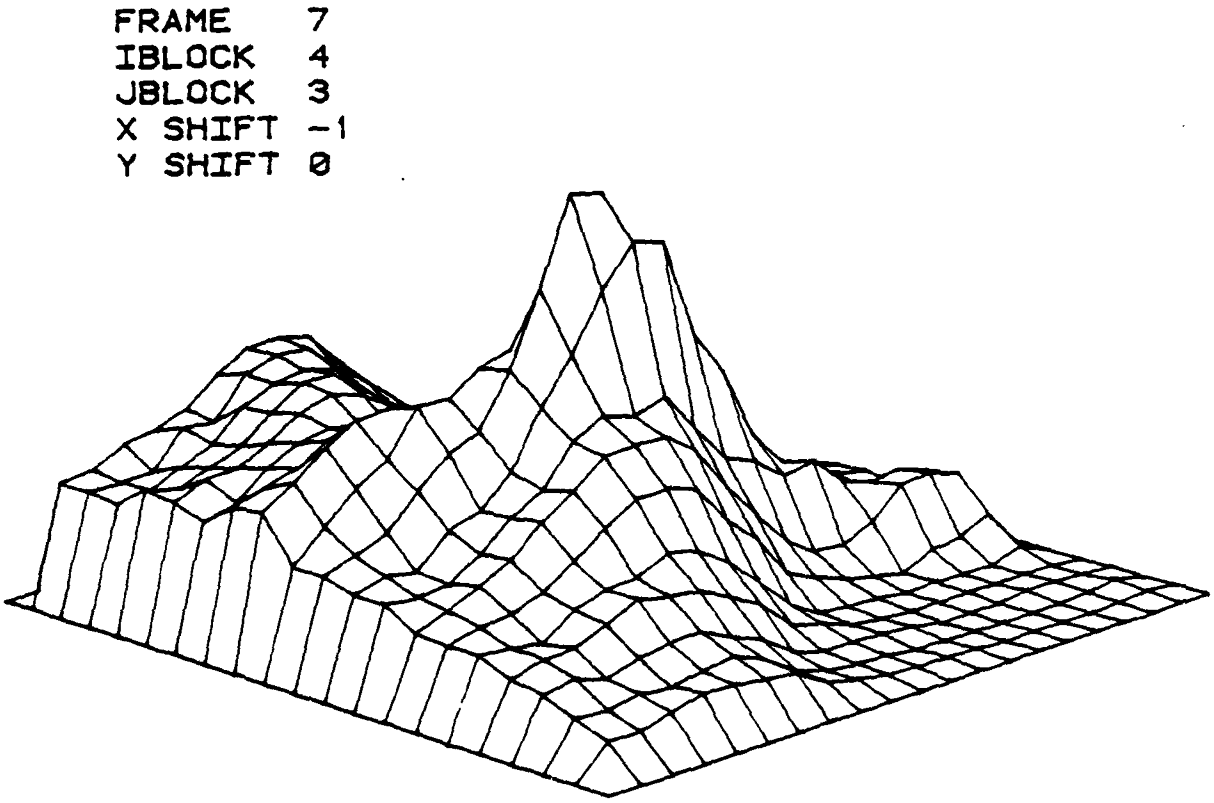


FIGURE 4.6(d) Plot of Actual Metric

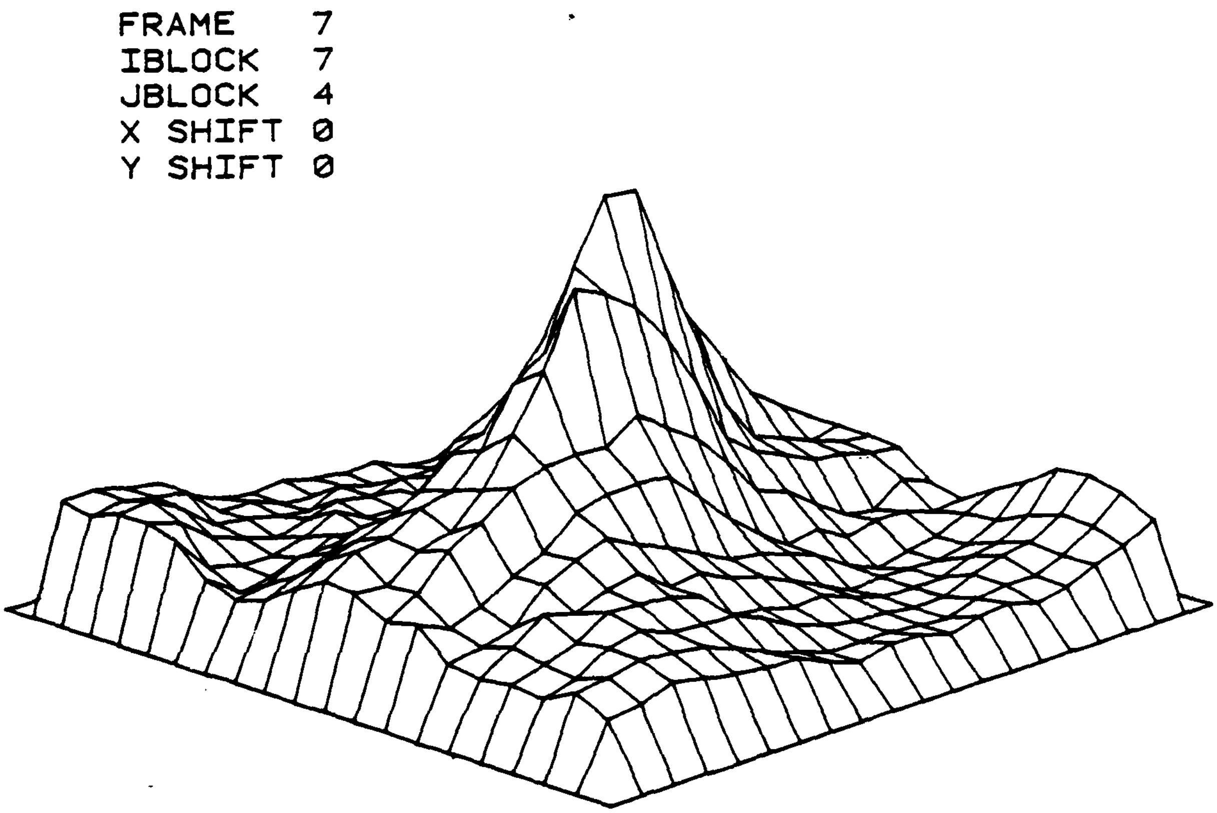


FIGURE 4.6(a) Plot of Actual Metric

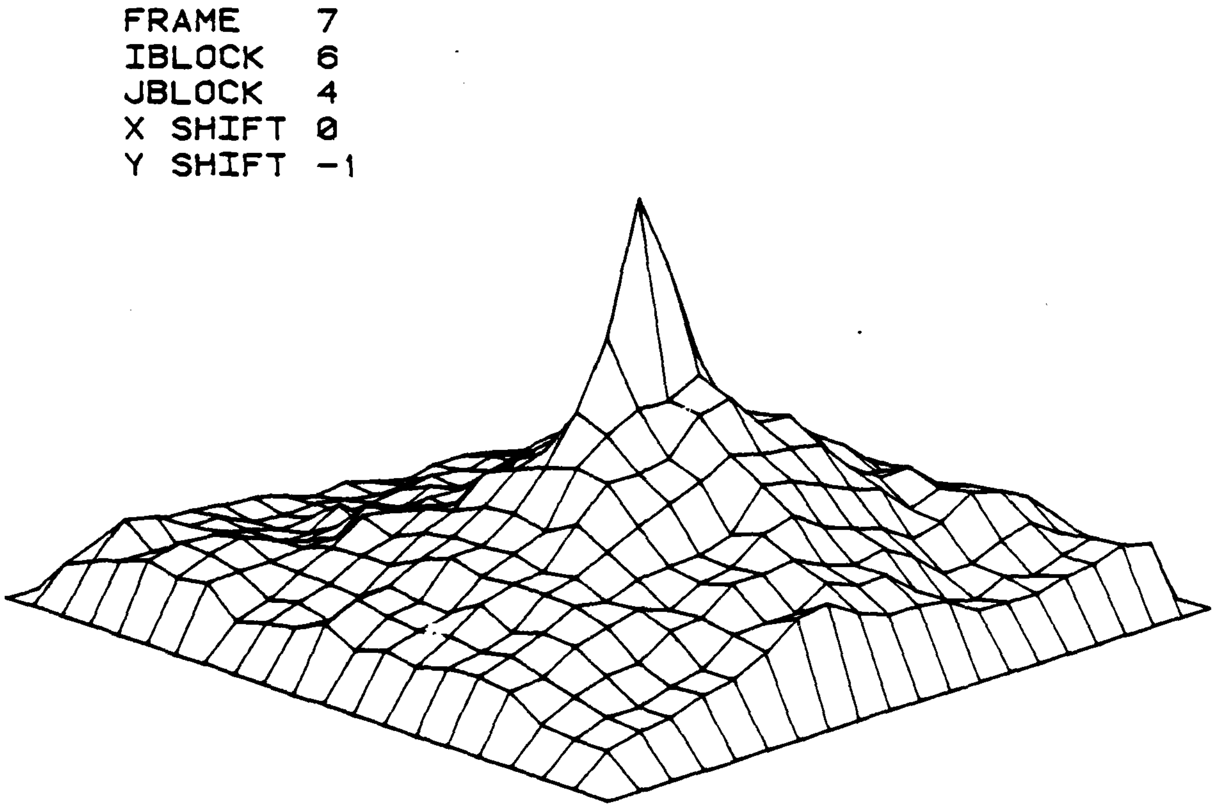


FIGURE 4.6(b) Plot of Actual Metric

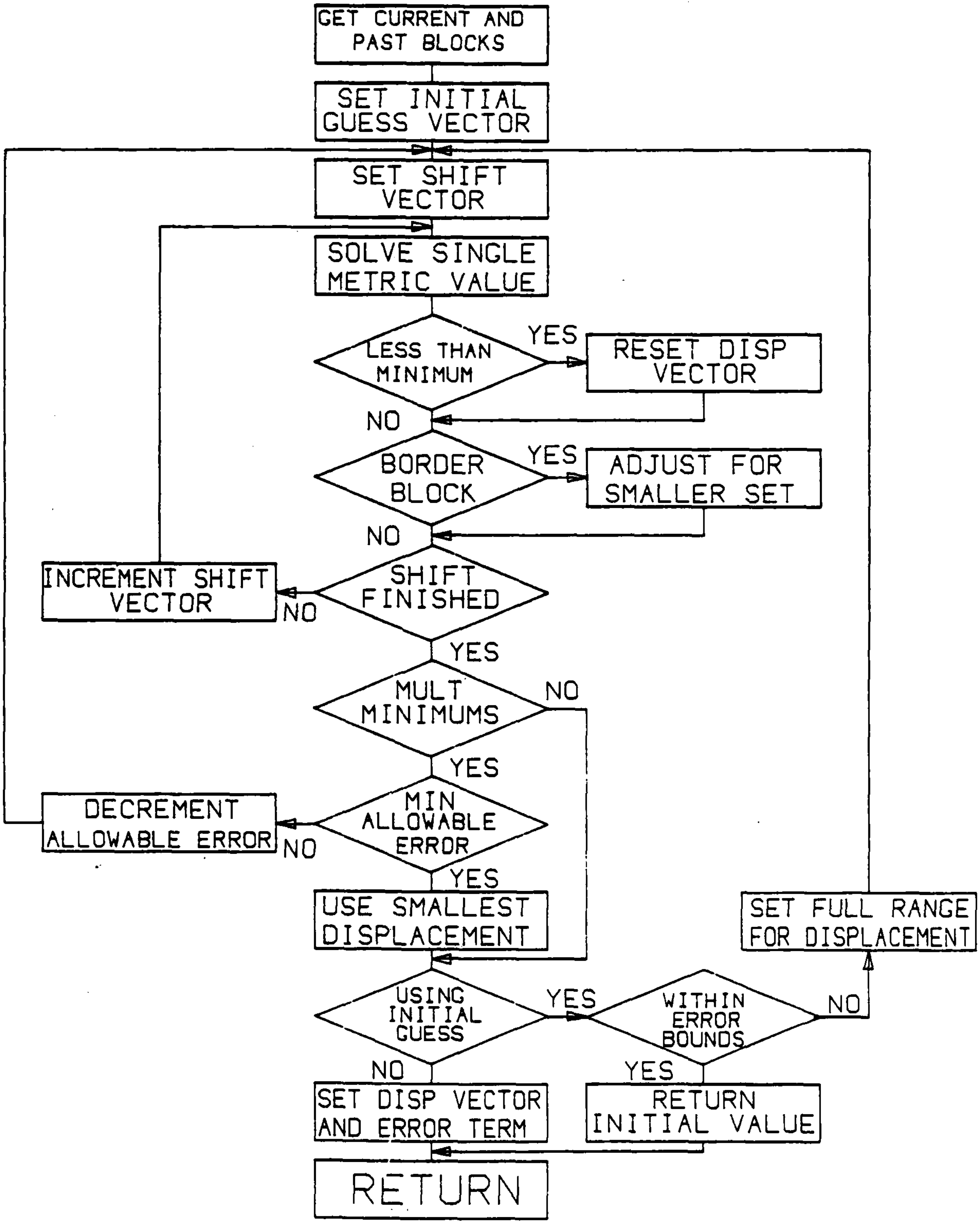
estimate of the displacement. This presentation is achieved with the use of subroutine INVERSE and MPLOT as given in appendix B. Figures 4.6 (a) through (d) are examples of various actual metrics. From these, the geometrical nature of the metric function can be seen.

The plots are arranged such that the base of each plot represents the value of the integer shift. Each base axis represents a direction of shift. The values of shift for these examples range from minus nine to plus nine in both the X and Y direction. The magnitude of the plot can be thought of as a measure of similarity between the current block and various shifted blocks from the previous frame.

difficult in practice. Following the flow chart of figure 4.5, the process starts with setting an initial guess for the displacement vector estimate. This is not used as starting point for iteration, but rather as a method to save time. This value can be set to zero, if no motion is expected, or to the value of the previous block estimate. The subroutine will test the hypothesis that this is the correct integer displacement, and if it falls within the threshold bounds, will exit the subroutine setting the actual estimate to the initial guess. If the metric value falls outside the threshold, the entire procedure must be executed for all allowable possibilities of block shift. The procedure is executed as follows. First the shift vector estimate must be set to the lowest possible value in both the X and Y directions. Solve for the metric or risk function based on this shift vector. Recall from Chapter 3 that the metric is defined to be a function of the sum of the thresholded differences between the current frame block and the previous frame pixel neighborhood. The shift vector is then incremented and a new metric value calculated. The procedure is continued until all allowable pixel displacements have been tried. At this point, a matrix of metric values, whose entry position defines the associated displacement estimate, is examined for a minimum value. The location of the minimum value designates the integer displacement estimate. If there is more than one minimum value, the entire metric matrix is recalculated using a smaller allowable error threshold. If after this is tried, and the allowable error cannot be decreased and there are multiple minimum points, the smallest displacement value is chosen to be the actual estimate. Along with the location of the smallest metric value, the actual value of the metric is also returned to the calling program as an aid in quality measurement. The program is now ready to return to the main driver program and continue.

Before continuing with the program procedure, it is informative to look at some examples of the metric matrices generated by the algorithm. Recall that the actual implementation looks for a minimum value of the metric matrix, but for ease of presentation the metric matrix has been modified. The modification is accomplished by reversing the magnitude order of the data and rescaling to fit the 0 to 1 range. The reverse in order exchanges places of the minima and maxima. Hence, in the presentation the maximum value of the function defines the best integer

FIGURE 4.5 Metric Subroutine Flow Chart



Following constant initialization, the single image is broken up into many small sub-images of size 16 by 16 and blockwise scanned top-to-bottom and left-to-right. The following discussion refers to operations on a single block.

First it must be determined if the particular block is a border block. That is, does it border the outside of the image? The reason for this is that border blocks request data from the previous reconstructed pixel neighborhood that do not exist. This is dictated by the possibility of a plus or minus p pixel shift in both directions. Rather than setting the nonavailable values to zero, it has proved advantageous to set these terms to the mean value of the remaining pixel neighborhood.

4.3 METRIC GENERATION

Given the current block and the previous pixel neighborhood, the next step is to calculate the difference metric. The flow chart for subroutine METRIC is provided in figure 4.5 while the subroutine listing is provided in appendix B.

Subroutine METRIC serves a dual purpose. It is used to determine the best estimate for the integer portion of the displacement based upon the arithmetic metric used. It is also used to determine if this integer estimate is a sufficiently good estimate. In the discussion to follow the absolute value threshold method, or MAP estimator given in section 3.3.3, will be used. In the subroutine, like the main driver, it is important to treat border and nonborder blocks differently. The execution is similar for both border and nonborder blocks, with the exception of a correction factor that takes into account the possibility of a smaller number of terms used in the border block calculations. The correction factor is used to weight the estimates based on fewer terms differently than it would if the entire previous data field had been available.

It is this portion of the algorithm that is the most calculation intensive, and hence the slowest to execute. Here again the algorithm, although implemented in series, lends itself to parallel operations. The implementation of the algorithm is simple in concept, but more

FIGURE 4.4 Main Control Program Flow Chart

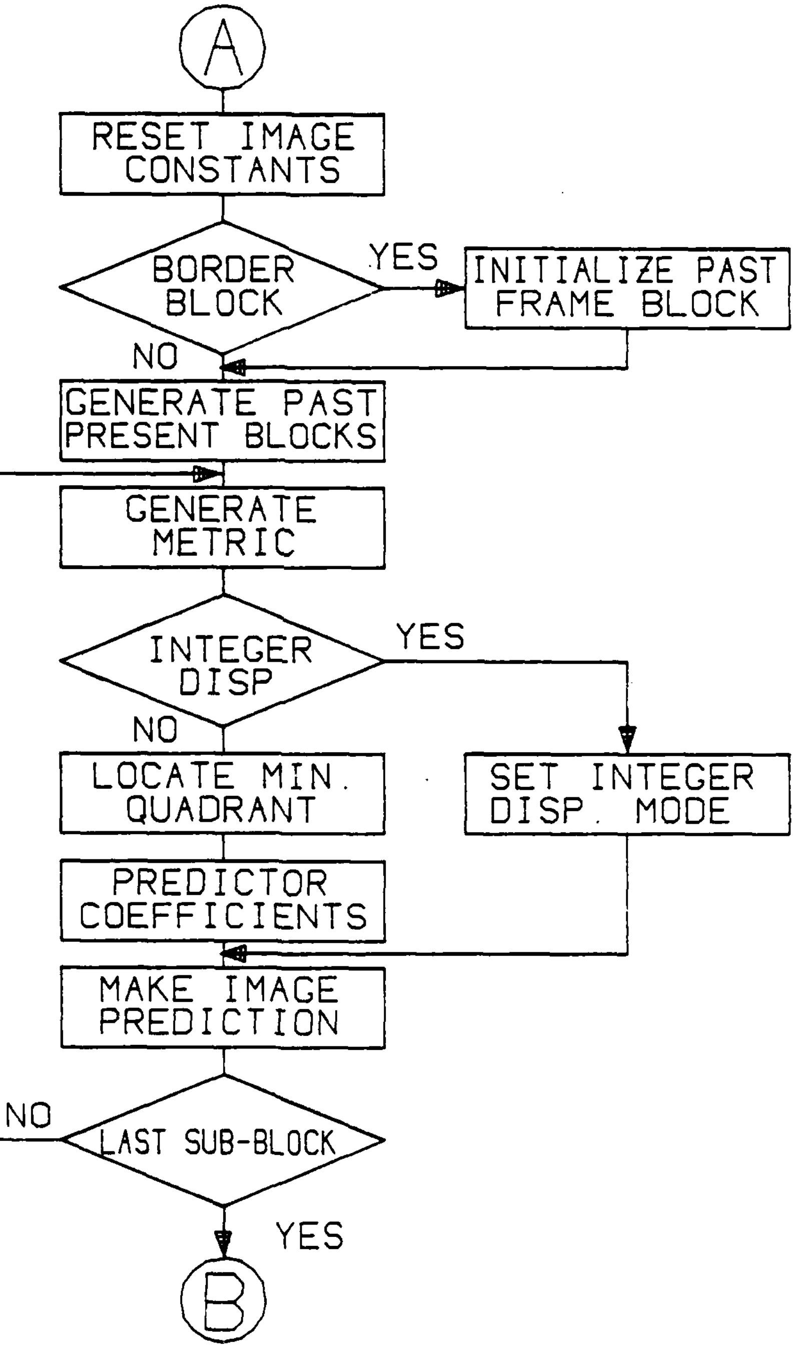
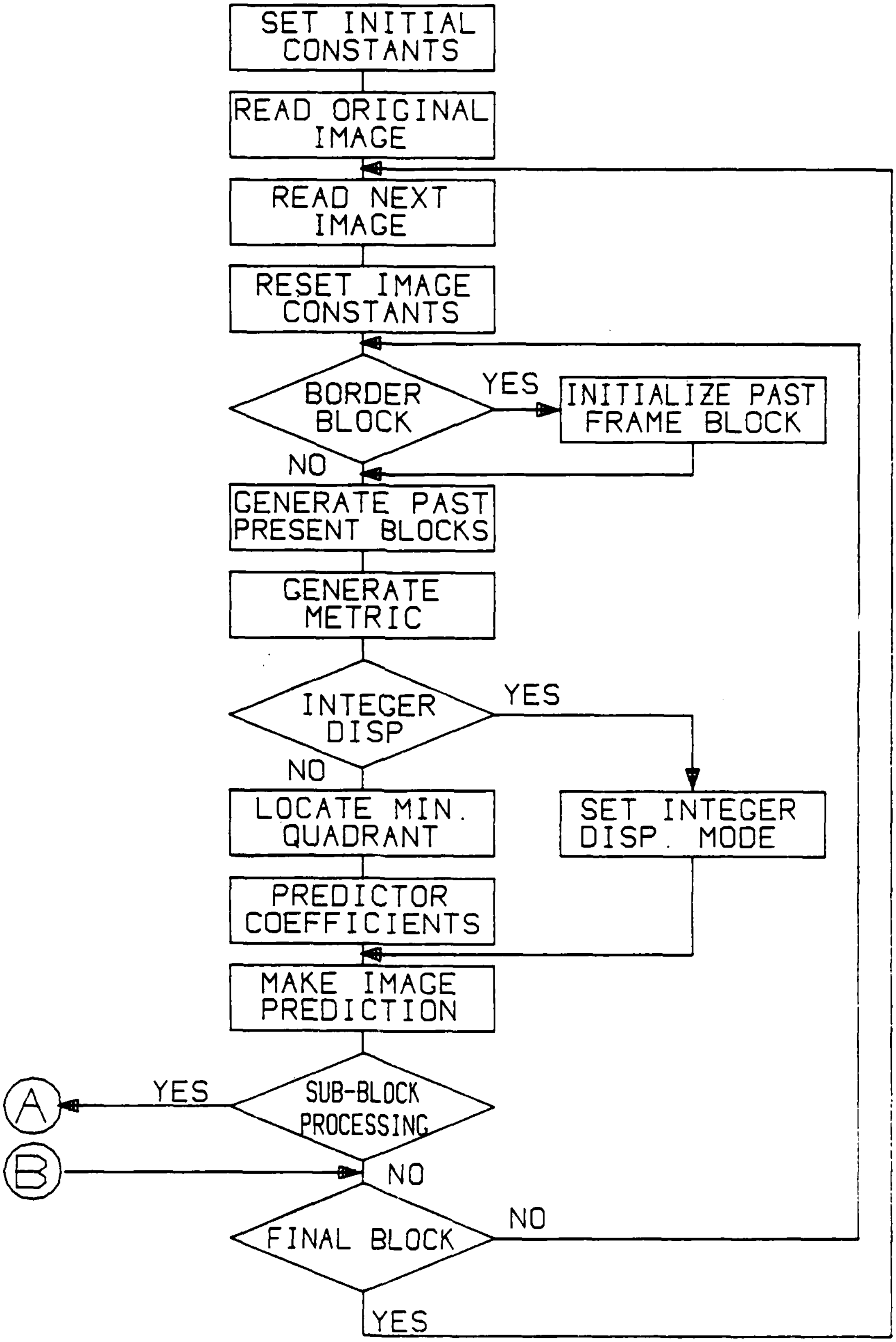


FIGURE 4.4 Main Control Program Flow Chart



4.1 ALGORITHM IMPLEMENTATION

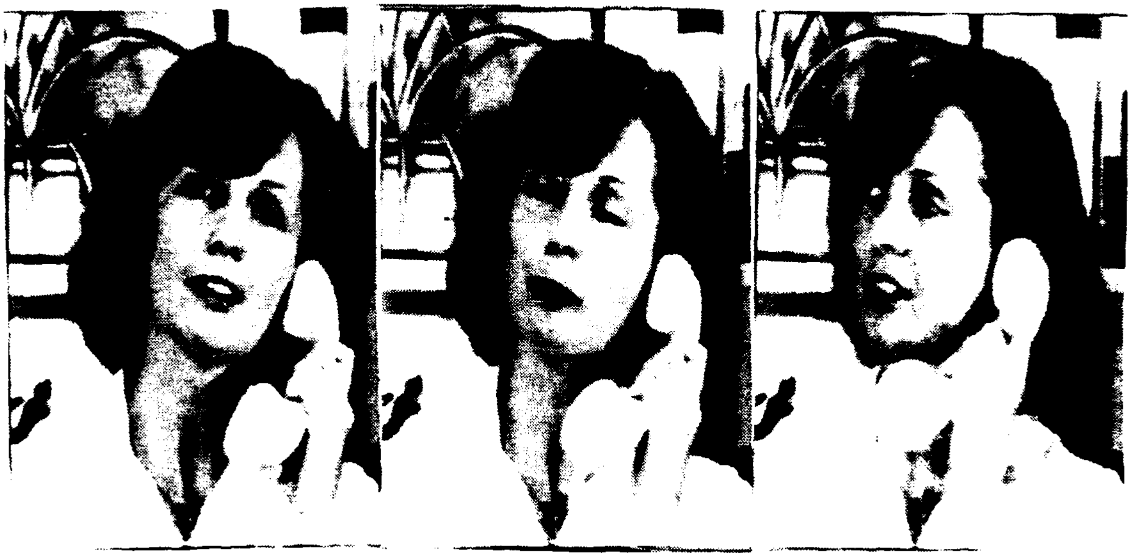
This method for motion compensated image coding lends itself very well to modular programming design. The system must perform serially in terms of most within block operations, but is easily adaptable to perform parallel or concurrent operations across blocks. Because the algorithm was modelled and simulated in software, only serial operations can be used and hence, the discussion to follow will be restricted to a purely serial implementation. Notes will indicate the parts of the implementation that are easily converted to parallel operations.

As noted earlier, it is assumed that both the transmitter and receiver have complete knowledge of the initial image and motion compensated image coding starts with the second frame of the sequence. The actual FORTRAN code used for the simulation is provided in appendix B and will be referred to throughout this chapter. Flowcharts for both the main program and some of the subroutines will be provided in the text as required.

4.2 MAIN CONTROL PROGRAM

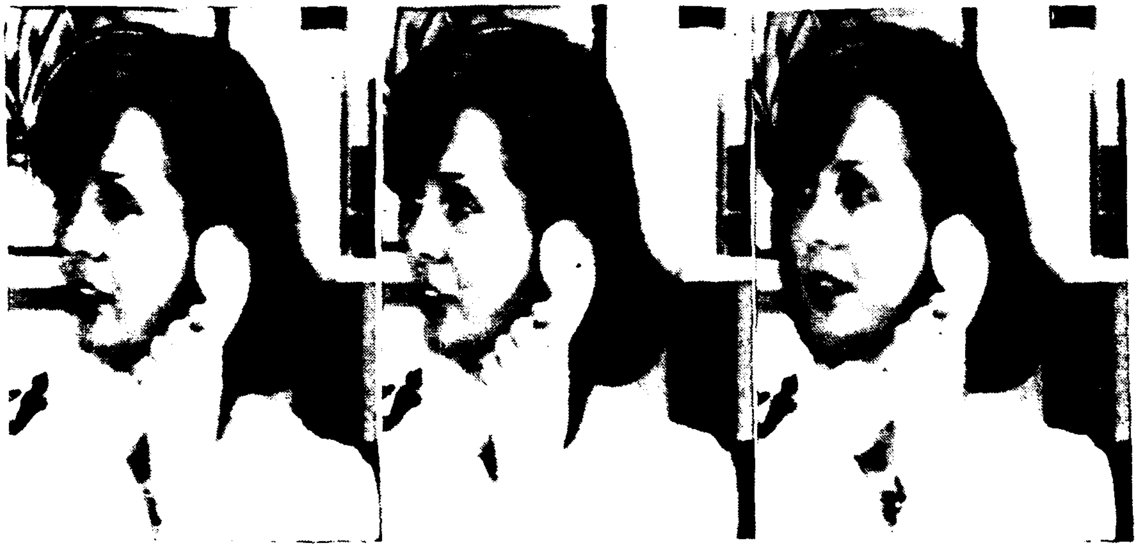
Figure 4.4 provides a flow chart of the main control program and relates to the program listing MAIN in appendix B. The program initialization section sets up the required run time constants used for both variable assignment and program control. Some of the more important constants will be discussed below.

NPBITS is an integer number used to designate the number of bits used to quantize the predictor coefficients. The quantization method assumes a uniform distribution from -2 to 2. The actual number of quantizer bits is one greater than NPBITS to allow for a sign bit. The variables VAREST, STDERR, and AVGERR are set constants used as threshold values throughout the program. VAREST is an estimate of the average difference between two identical frames when viewed at the output of the imaging device, or in other words an estimate of the imager noise. STDERR and AVGERR are threshold values for the error terms. ISIZE and JSIZE are simply used to identify the image size and may be changed to fit any image size as long as the DIMENSION statement settings are likewise adjusted.

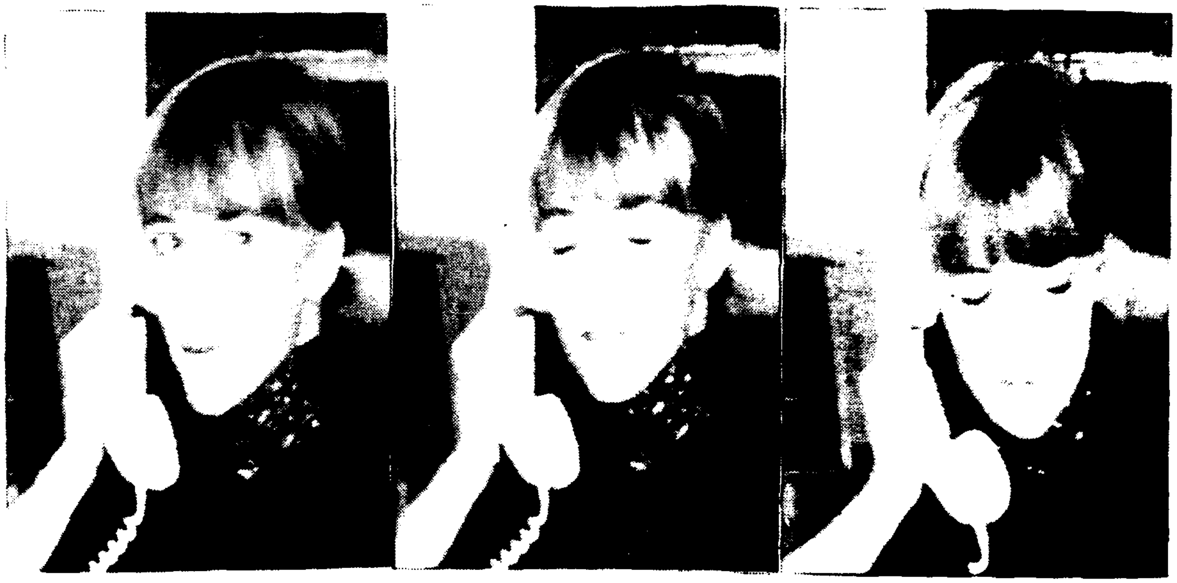


(a) FRAME 1 (b) FRAME 8 (c) FRAME 15

Figure 4.3
Original 'FAST PHONE' Sequence

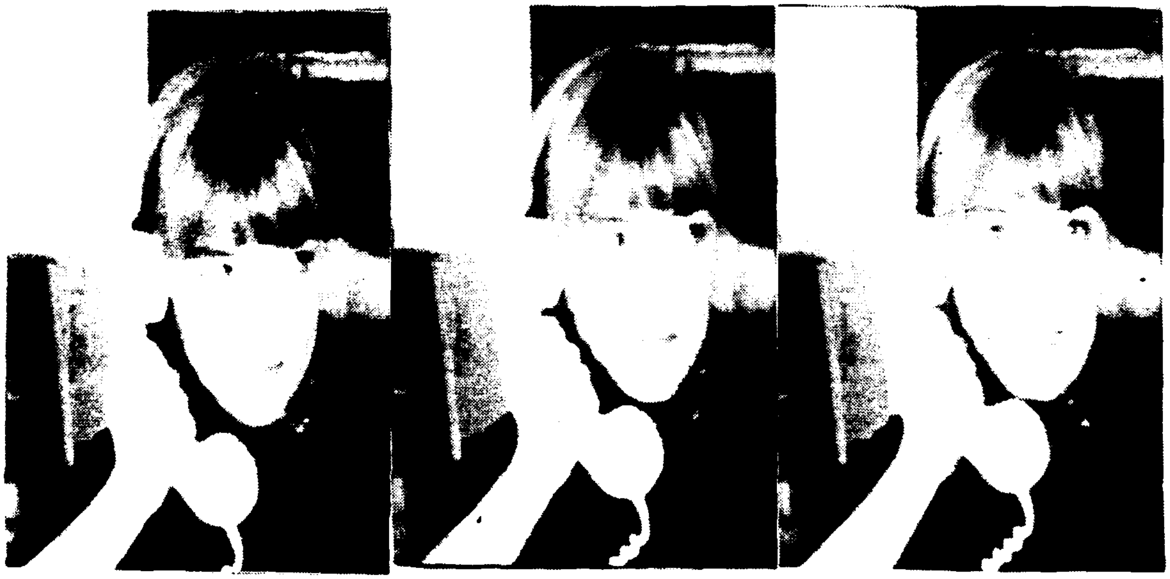


(d) FRAME 22 (e) FRAME 29 (f) FRAME 36



(a) FRAME 1 (b) FRAME 8 (c) FRAME 15

Figure 4.2
Original 'SLOW PHONE' Sequence



(d) FRAME 22 (e) FRAME 29 (f) FRAME 36

viewed in sequence because of the integrating ability of the eye. For simple comparison purposes, six frames from both of the original sequences will be presented. Figures 4.2(a) through (f) are frames from the 'SLOW PHONE' sequence and 4.3(a) through (f) are from the 'FAST PHONE' sequence. It is important to note that much of the image quality is lost in the photographic process and the associated copying and printing.

The theoretical system presented in Chapter 3 has been programmed and simulated using FORTRAN and hence was not executed in hardware or real time. More on the actual implementation of the algorithm will be presented in the later sections.

The algorithm simulation starts with the assumption that both the transmitter and receiver have full knowledge of the first frame. The following frame starts the process of motion compensated image coding. Transmission of the first frame may be accomplished using normal coding techniques or transmitted using ordinary PCM methods. The process from this point onward assumes that one frame is read at a time and processed before the next frame can be read. Although none is used in this simulation, it is understood that in a real world implementation a fixed length buffer would have to be incorporated into the hardware. It is assumed that an infinite length buffer is available in the simulation. Although not used for buffer control, a very good built in feature of the implementation is the setting of a required output signal to noise ratio. If a fixed length buffer were used, the varying of this required SNR could function as a buffer control parameter. That is, when the buffer approaches full the required SNR could be allowed to drop. When the buffer neared empty, the output SNR could then be increased. This would nearly eliminate the possibility of a buffer overflow, and hence a loss of data.

4.5 REGRESSION COEFFICIENT GENERATION

The generation of the delta function magnitudes is based on linear regression theory and is accomplished in the subroutine COEFGN. The magnitudes of the delta functions are used as weighting values to estimate the current frame from the past frame. Given the integer displacement value and the magnitudes of each of the delta functions, the system is able to make a good prediction of the current block.

Because the displacement has been bounded, the four pixels that are used to estimate the current pixel are known. What is not known, is how these pixel intensities are combined to produce the current estimate. The weighting values define how the previous pixels are to be combined and are determined by solving for the parameters of the autoregressive spatial-time series.

4.6 CURRENT BLOCK GENERATION

Subroutine PRED is the portion of the program that is used to calculate the current block estimate. Along with this estimate, an error block is also calculated. From the error block, the error mean and variance must be determined. The mean and variance are used as an indication of how well the prediction process performed. If both the mean and variance are within set limits, the process will simply go on to the next block. If the error is deemed too large, the entire process up to this point is redone on a smaller blocksize of 8 by 8. This says that the single 16 by 16 block is broken up into 4 sub-blocks and the algorithm is executed for each of the sub-blocks. If the blocksize is already 8 by 8 and the error is still too large, the process must generate a quantized error or quantized residual term.

The error in prediction can come about from many sources, some of which are listed below. The motion that occurs may not fall into the category of pure translation. That is, the motion may consist of various types of rotation. The objects could have rotated in the plane of view or in any of the planes perpendicular to the plane of view, hence taking into account the 3-D aspect of the objects. The model cannot account for rotation. Hence, when this occurs errors will result that must be quantized and transmitted. The other major source of error generation is from the problem of background/foreground interaction. Recall that the motion model was developed for the pure translation, no background case and hence only optimally estimates this type of motion. When the actual data does not fit the model, it may be necessary to quantize and transmit the error.

4.7 ERROR QUANTIZATION

The error term resulting from the subtraction of the predicted block from the actual block is very scene dependent. Because of this the statistics for the error block are widely varied. The quantizer used for the residual encoding is very important because it determines, to a large extent, the overall system data rate. The variance of the error is a good indicator of image quality. That is, if the variance is low the image reproduction should be good. On the other hand, if the variance of the error is large, it is expected that the reproduced image will be somewhat degraded. It only seems logical, from the above discussion, that the quantizer used should be able to adapt to the changes in statistics of the error and transmit only the data required to reach some fidelity criterion.

Subroutines QUANTI and QUANTZ are used in the quantization of the error signal. QUANTI determines the amount of gain the quantizer produces for the various number of output levels. QUANTZ performs the actual quantization and coding of the residual.

Chapter V
RESULTS AND CONCLUSIONS

The presentation of results from time sequential image processing, and even image processing in general, proves to be a difficult task. When a human viewer is the final link in this imaging system, the important criterion is not a set of numbers calculated from various system parameters, but rather the important criteria is how does the viewer subjectively judge the quality of the final output. Although there appears to be some correlation between some distortion measures, such as the mean square error, and the subjective quality, this correlation is not very strong. Another way to put this is, a decrease in the mean square error may not always mean an increase in judged subjective quality.

A presentation of output results of single images, not taken from an image sequence, poses a simpler problem than those taken from image sequences. Each image can be viewed subjectively apart from all others because they are not dependent on images that come both before and after in time. This method of presentation does not hold, as well as is not feasible, for sequential image data. When the sequences are composed of thirty or more frames per second, the amount of space required to present them in itself creates a problem. But more importantly, the eye and associated biological processors prefer that the images be presented in a sequential manner in the same spatial location. It appears that the integrating ability of the eye plays an important role in the subjective image judgement of time sequential imagery.

The output images will be presented in a fashion similar to the presentation of the input images provided in chapter four. One should note that quality degradation occurs at every step of the image transfer process. That is, exposure, development, printing and copying all degrade the true quality of the actual image.

Even though distortion measures such as the mean square error may not be directly related to the subjective quality of the output image, it is an indicator that allows for a comparison among different runs of the same program with varied parameters. It does function as a means to compare outputs on a more numerical level. It seems logical that if the system and algorithm remain the same, with only minor modifications in some system parameters, that the outputs can be compared on a more quantitative basis. Stating this simply, the image sequence with the best overall output signal to noise ratio should be the image sequence judged best in a subjective analysis.

Another problem in the presentation of results arises from the very large variation in possible input image sequences. Because the data compression that occurs in the system is a direct result of the exploitation of the between frame correlation, the amount of data compression that can occur is directly related to how well adjacent frames are correlated. It has been shown in intraframe image coding techniques [8], that the majority of the information of the image is contained in the edges of the image. Along with the majority of the information, the edges also require the majority of the bandwidth in a data compression scheme. The same can be said for interframe image coding if one defines what is meant by 'edge' in interframe terms. If an intraframe edge is defined to be a boundary between two regions of near uniform luminance and can be estimated by using a spatial derivative, the interframe edge can be defined as a boundary between spatial regions of varying intensity temporally separated and can be estimated with a temporal derivative. What this says is that the edge, in interframe terms, occurs at the portions of the image that change between frames. This idea can be further restricted if the method of interframe image coding is motion compensated. Now the edge can be defined to occur at the regions of the image that interface the background and foreground. Hence, if this interframe edge were viewed, the edges would appear at points where the moving objects and background meet. Just as in the intraframe case, the temporal edge requires the largest amount of bandwidth for transmission, and hence contains the majority of the new information contained in each image.

Because the amount of information that must be transmitted is directly related to the quantity of temporal edges and hence motion, the amount of motion that occurs in a given image sequence should provide a good indication of the data rate requirements of the system.

For presentation purposes, only a fraction of the total number of images can be presented. Figures 5.1(a) through (f) are a selected set of output images from the 'SLOW PHONE' sequence with the following set of selected parameters. These parameters will be changed to try and show the effects of the various thresholds and quality settings on the quality of the output.

```
NPBITS = 10
VAREST = 1.0
STDERR = 3.0
AVGERR = 3.0
SNRSET = 24.0
```

Figures 5.2(a) through (f) are also from the 'SLOW PHONE' sequence with the following set of selected parameters.

```
NPBITS = 10
VAREST = 1.5
STDERR = 4.0
AVGERR = 4.0
SNRSET = 21.0
```

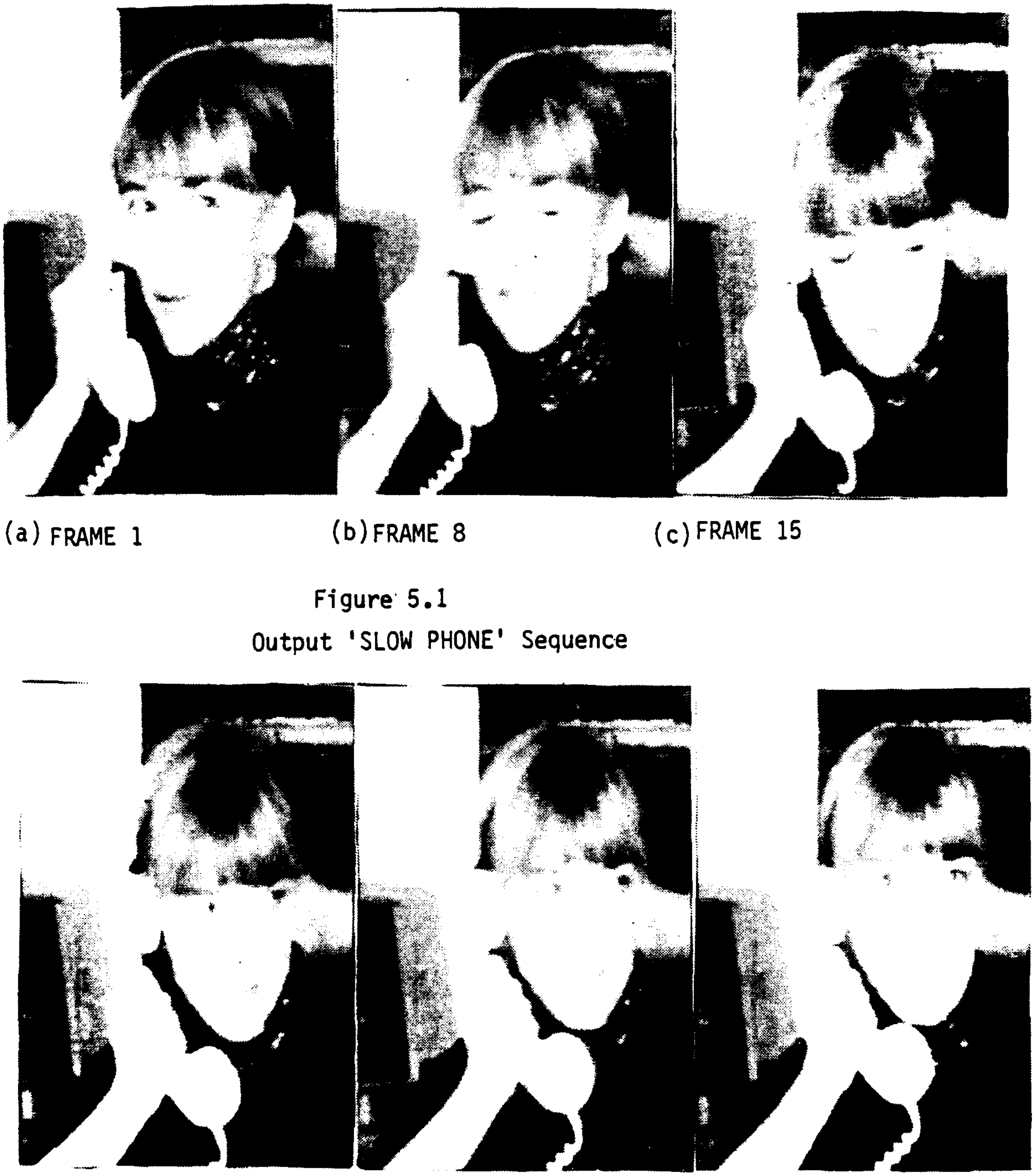
Figures 5.3(a) through (f) are selected images from the 'SLOW PHONE' sequence with the following set of selected parameters.

```
NPBITS = 10
VAREST = 1.0
STDERR = 2.0
AVGERR = 2.0
SNRSET = 27.0
```

Figures 5.4(a) through (f) are also images from the 'FAST PHONE' sequence with the following set of selected parameters.

```
NPBITS = 10
VAREST = 1.0
STDERR = 3.0
AVGERR = 3.0
SNRSET = 24.
```

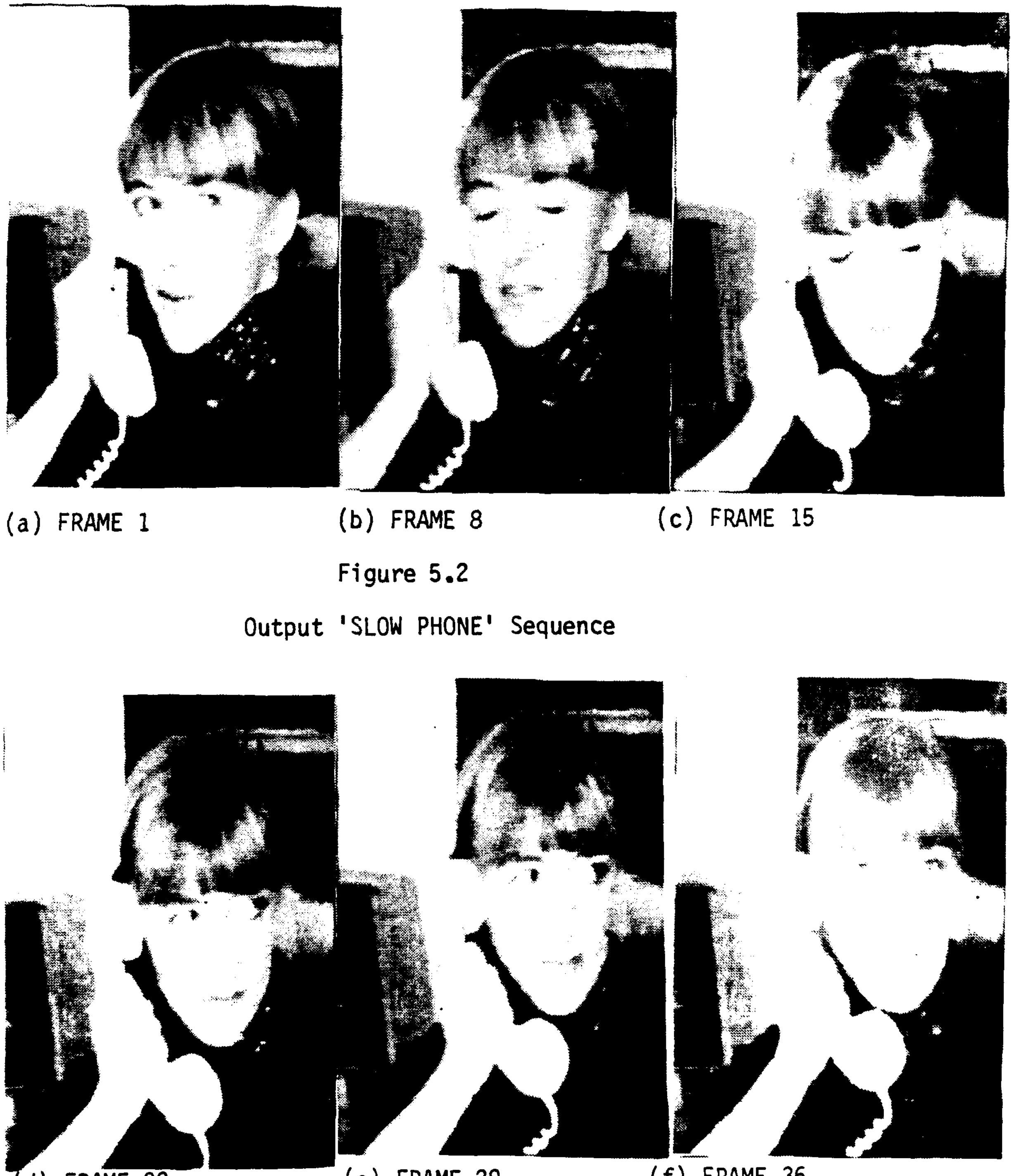
As stated earlier, the most important aspect of the output is the subjective quality of the images, but other parameters of the system may provide insight into functioning of the algorithm. For this reason, a set of eleven plots for each of the four output sequences will be presented. The plots will be presented in groups of four, each one



(a) FRAME 1 (b) FRAME 8 (c) FRAME 15

(d) FRAME 22 (e) FRAME 29 (f) FRAME 36

Figure 5.1
Output 'SLOW PHONE' Sequence

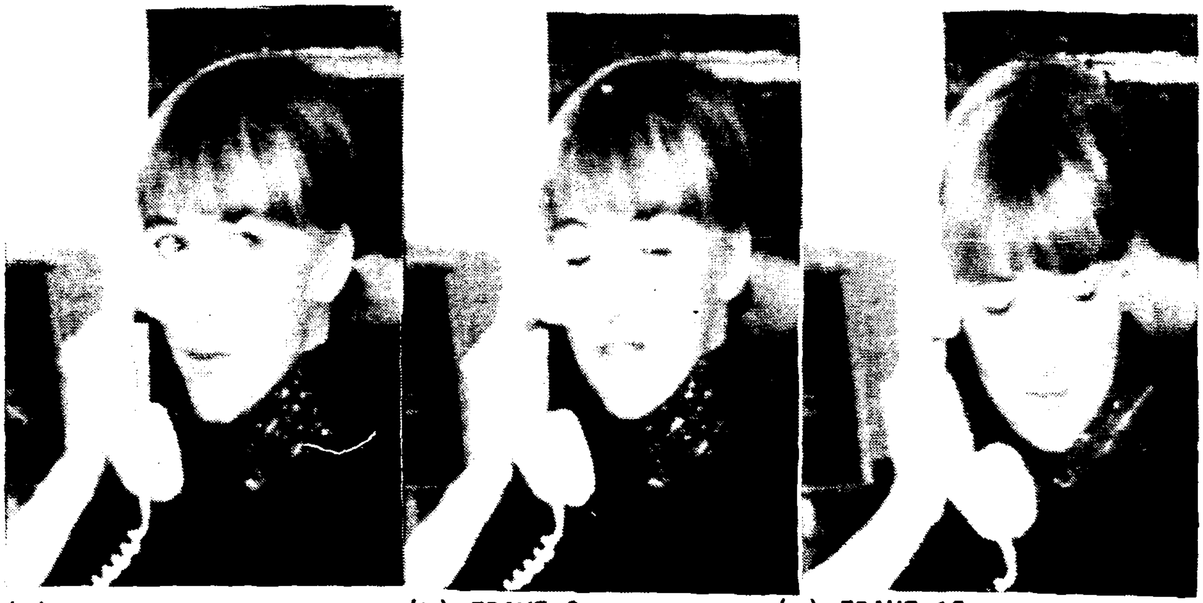


(a) FRAME 1 (b) FRAME 8 (c) FRAME 15

(d) FRAME 22 (e) FRAME 29 (f) FRAME 36

Figure 5.2

Output 'SLOW PHONE' Sequence



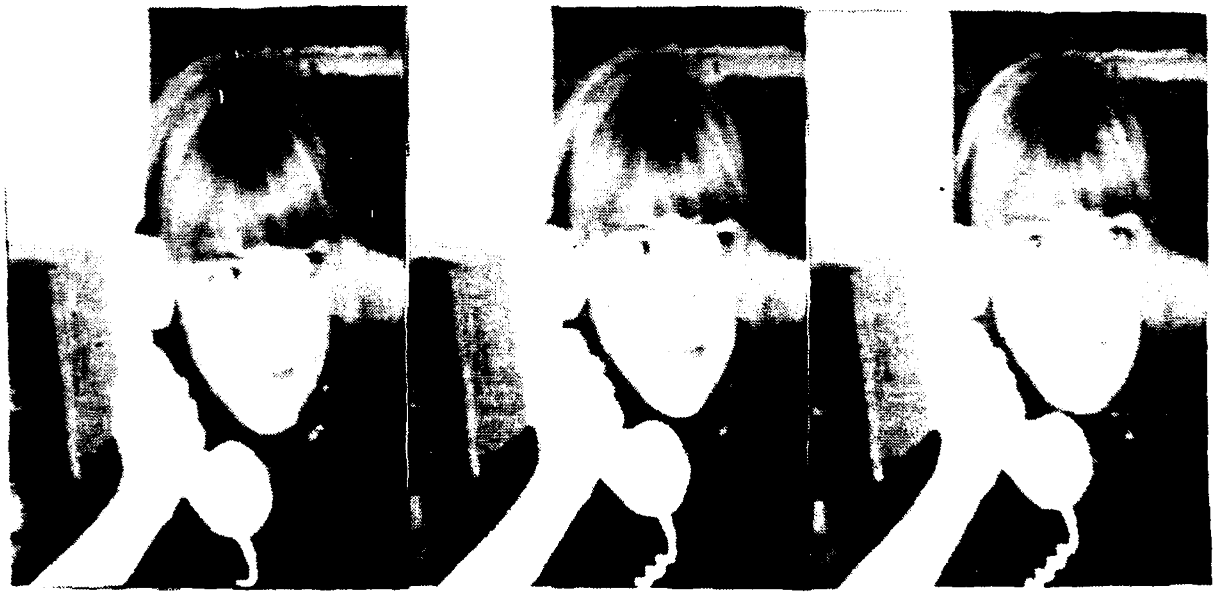
(a) FRAME 1

(b) FRAME 8

(c) FRAME 15

Figure 5.3

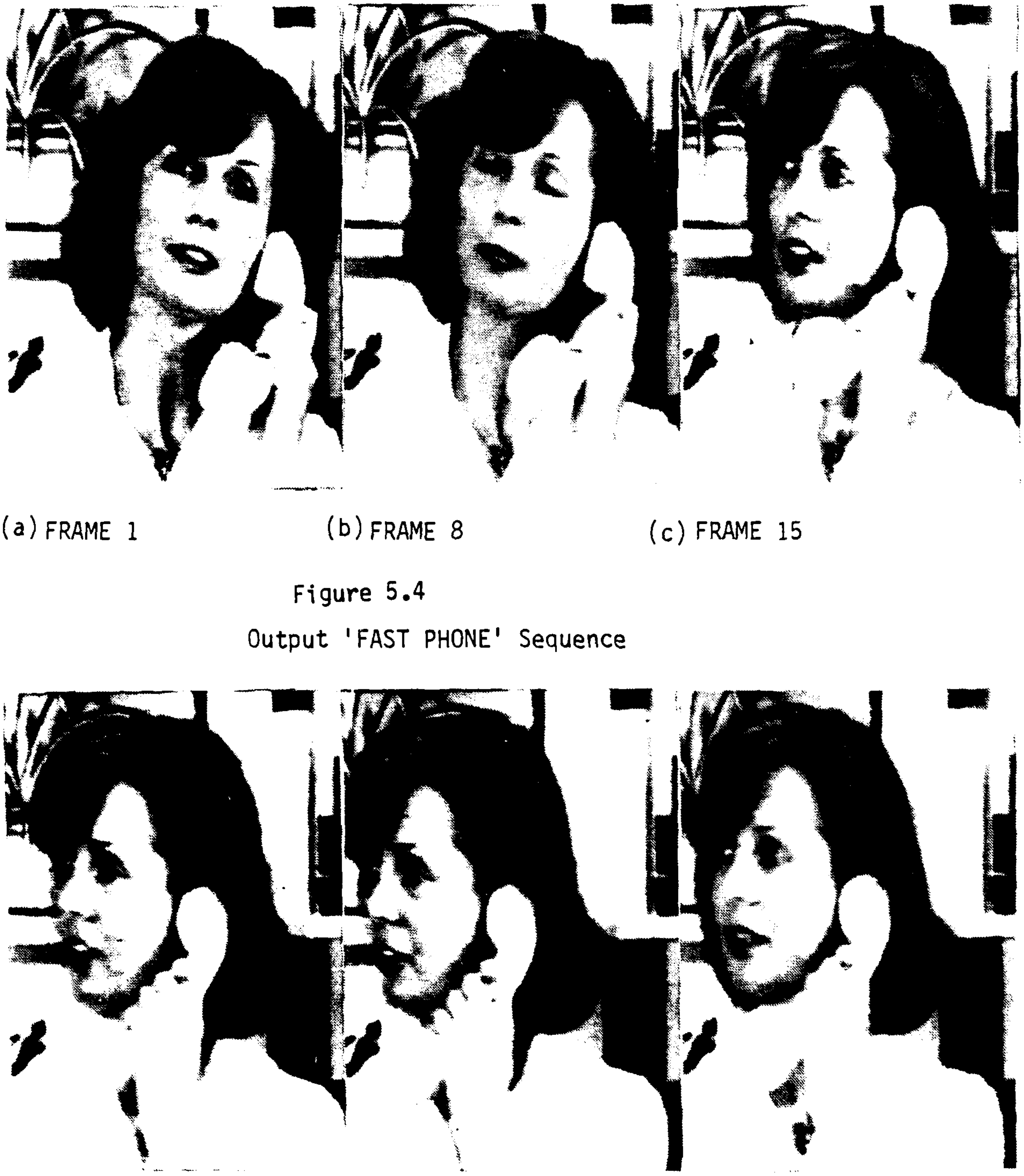
Output 'SLOW PHONE' Sequence



(d) FRAME 22

(e) FRAME 29

(f) FRAME 36



(a) FRAME 1 (b) FRAME 8 (c) FRAME 15

Figure 5.4
Output 'FAST PHONE' Sequence

(d) FRAME 22 (e) FRAME 29 (f) FRAME 36

plotting the same parameter as the others in the group, but with a different image set or initial parameter setting.

The first set of plots, figures 5.5(a) through (d), provide a measure of the amount of motion that is present in the image sequence. For the purpose of the plots, motion is defined to have occurred within a block if any information about that block must be transmitted. This includes cases of integer-only displacement, non-integer displacement and possible cases where no motion has occurred, but a residual data sequence is required. For counting purposes, a block is defined as an eight by eight sub-image. The results are plotted as a percentage of the maximum number of sub-blocks possible. For the image size used, with an eight by eight blocksize, the total number possible is 384. The average number of blocks considered to be in motion in the sequence of forty images is printed in the upper right-hand corner of each figure. Here, as in the remaining ten plot sets, the (a) plot relates to the output figures provided in figures 5.1(a) through (f). Plots labelled (b), (c), and (d) then relate to figures 5.2(a) through (f) to 5.4(a) through (f) respectively.

The second set of plots, figures 5.6(a) through (d), present the instantaneous data rate as a function of frame number. Like the previous set of plots for the block rate, the plots start from a small value and quickly rise to a more stable value. This can be partially explained by noting that it may take multiple frames to detect and correct for very small amounts of motion between frames.

The next set of plots, figures 5.7(a) through (d), simply combine the two previous sets of plots in such a way so that the interaction of the block rate and data rate can be more easily seen. As would be expected, the points tend to cluster about a line angled from the lower left to the upper right. This simply shows that an increase in motion will require an increase in data rate.

Figures 5.8(a) through (d) provide a numerical indication of instantaneous system distortion. When it comes to presentation of results, not even the term 'SIGNAL TO NOISE RATIO' has a universal definition. As used here, signal to noise ratio is defined to be ten times the log base 10 of the ratio of the signal variance to the error variance. The average is again printed in the upper right-hand corner of each plot. Note the broken Y axis for signal to noise ratio.

Figures 5.9(a) through (d) and 5.10(a) through (d) are presented to show the interaction of the system gain, or signal to noise ratio, to both the block rate and data rate. For these sets of plots, it would be ideal if the system signal to noise ratio was not a function of data rate or block rate. The ideal case would be one for which the output signal to noise ratio would be constant regardless of the data rate or block rate. That is, the quality of the output images should remain constant regardless of what the input sequence contains. Neither of the sets of plots show a tendency to disprove this idea.

The previous three sets of plots were functions of the overall system gain. One of the internal parameters not measurable from the output of the system that has proved informative, is the predictor gain. The predictor gain is defined to be, ten log base 10 of the ratio of the variance of the signal to the variance of the error of prediction. This prediction error of the system is that error which would be generated if no residual were available for correction. Recall that this is one of the internal parameters used to judge what the data rate of the residual should be. The plots of the predictor gain as a function of frame number are provided in figure 5.11(a) through (d). Note that these plots tend to be much more erratic and varying than do those of the entire system signal to noise ratio. Although not plotted, the difference between system gain and the predictor gain shows the gain that has to be generated by the quantizer and error coding section of the system.

Unlike the overall system gain, it is expected that the predictor gain would be related to both the data rate and block rate. Figures 5.12(a) through (d) show the relationship between the predictor gain and the block rate. As would be expected, the points tend to cluster about a line with a slightly negative slope. Said another way, the predictor gain is higher for a smaller number of blocks in motion. The greater the number of blocks in motion, the lower the predictor gain. Likewise the predictor gain and system data rate exhibit similar characteristics in figures 5.13(a) through (d). The lower the predictor gain, the more data that has to be transmitted through the quantizer and hence, the higher data rates.

The final plots provide a sort of statistical view of the predictor error signal. Figures 5.14(a) through (d) present a log histogram of the prediction errors. Figures 5.15(a) through (d) provide an alternative view of the prediction error in a log cumulative distribution function of the error.

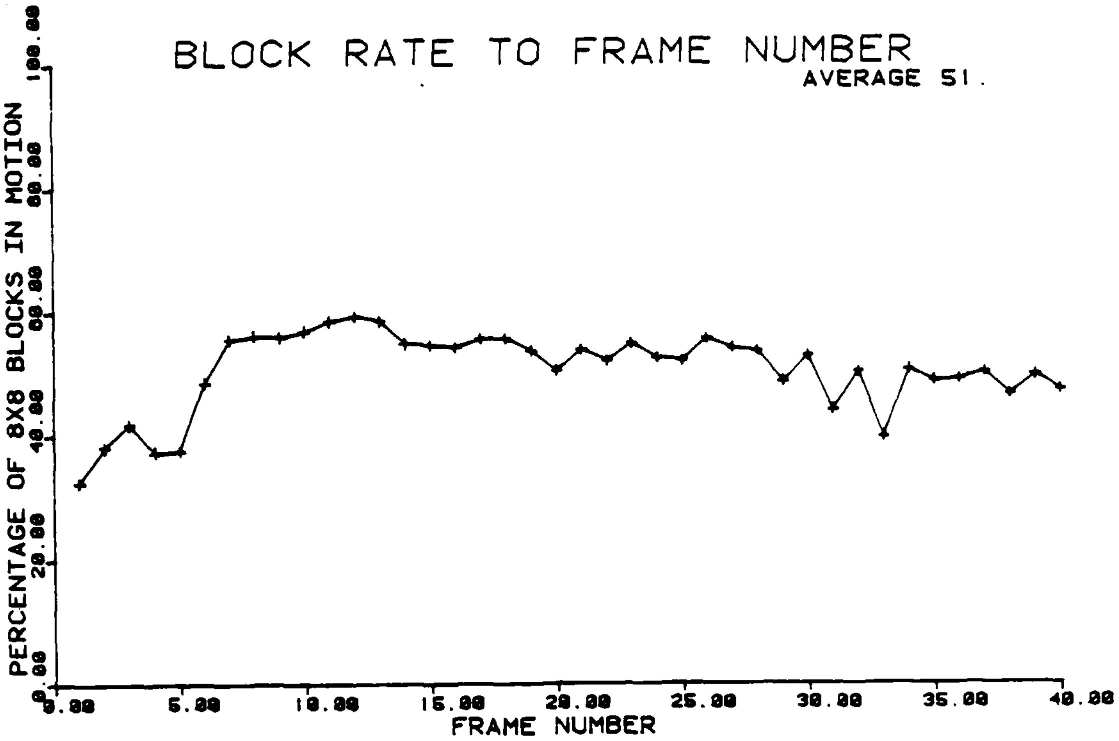


FIGURE 5.5(a) SLOW PHONE 1. Block Rate

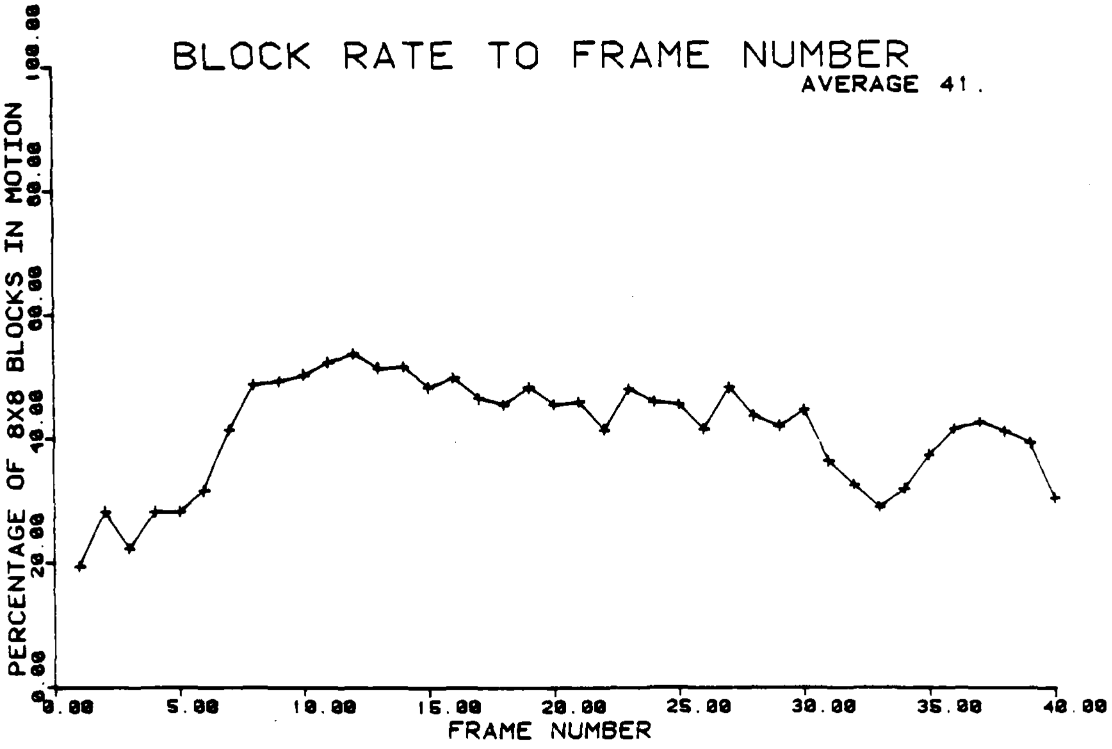


FIGURE 5.5(b) SLOW PHONE 2. Block Rate

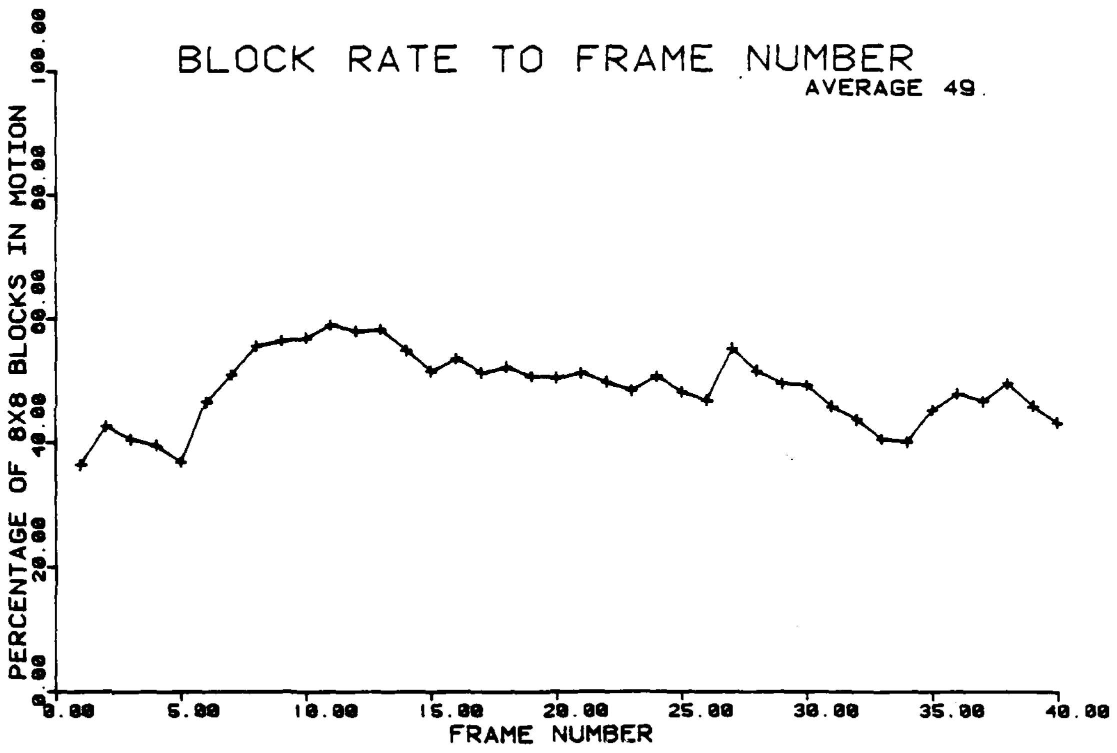


FIGURE 5.5(c) SLOW PHONE 3. Block Rate

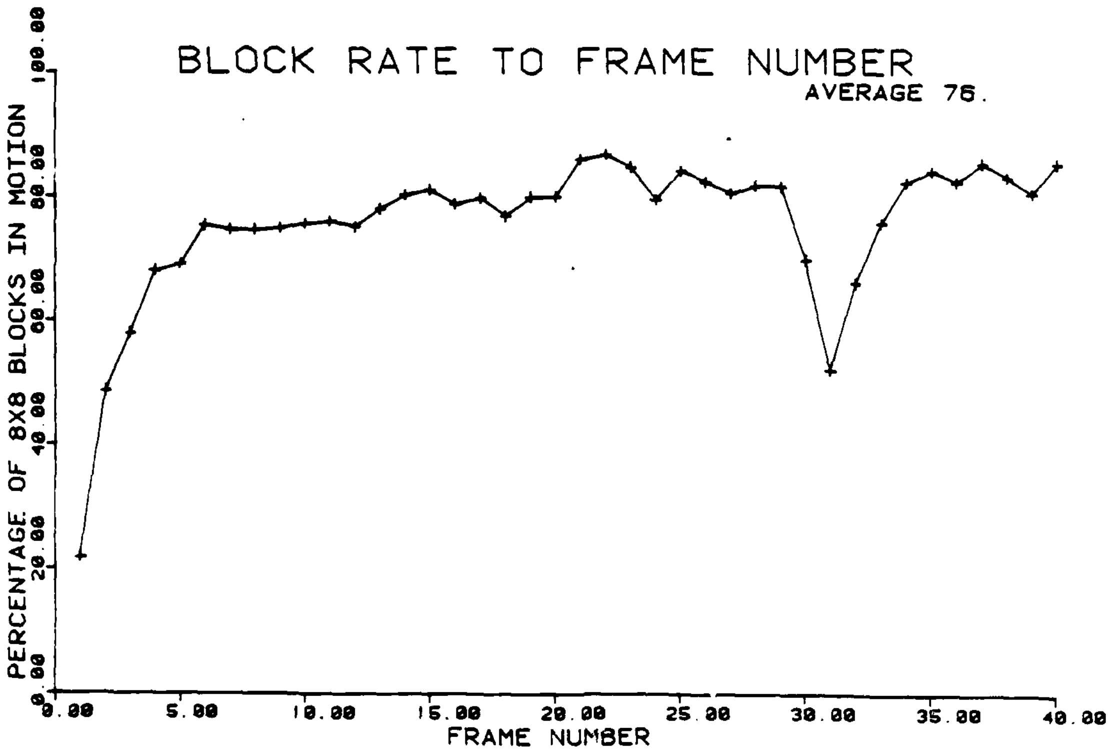


FIGURE 5.5(d) FAST PHONE. Block Rate

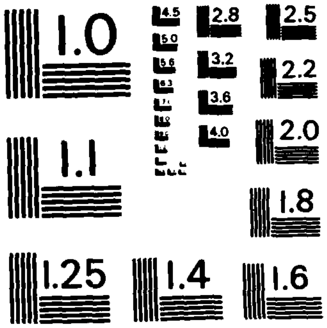
AD-A160 316 ADAPTIVE HYBRID PICTURE CODING VOLUME 1(U) ARKANSAS UNIV FAYETTEVILLE DEPT OF ELECTRICAL ENGINEERING R A JONES ET AL. 01 FEB 85 AFOSR-TR-85-0740 2/2

UNCLASSIFIED AFOSR-82-0351 F/G 9/2 NL

END

FILMED

DTIC


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

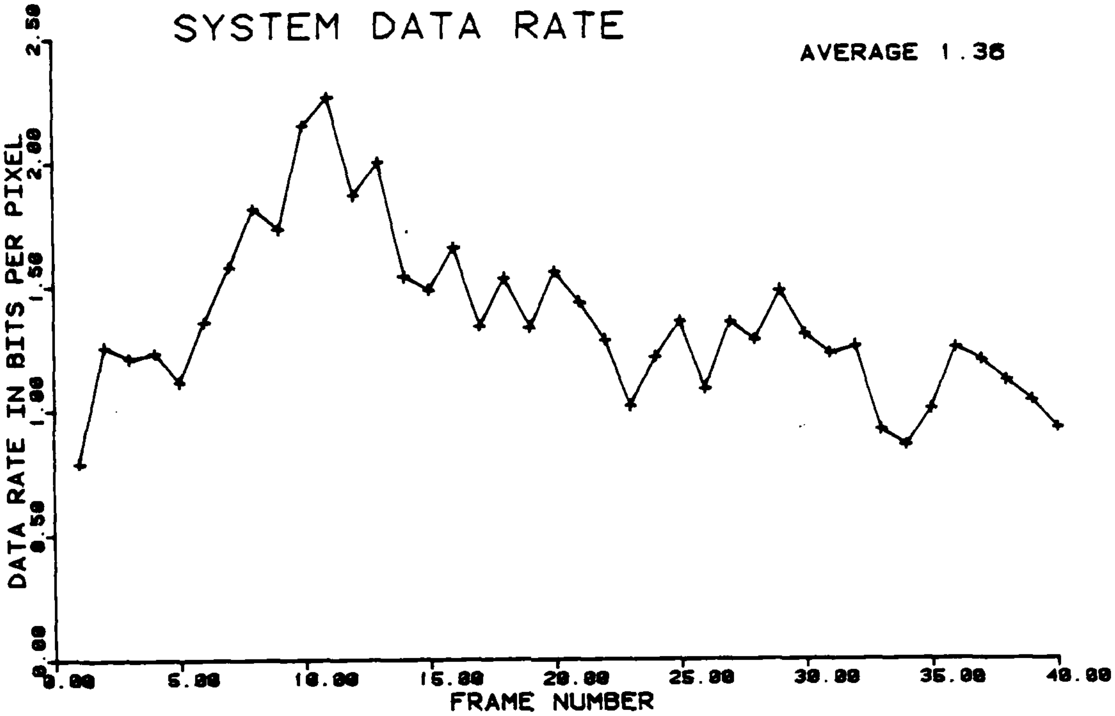


FIGURE 5.6(c) SLOW PHONE 3. Data Rate

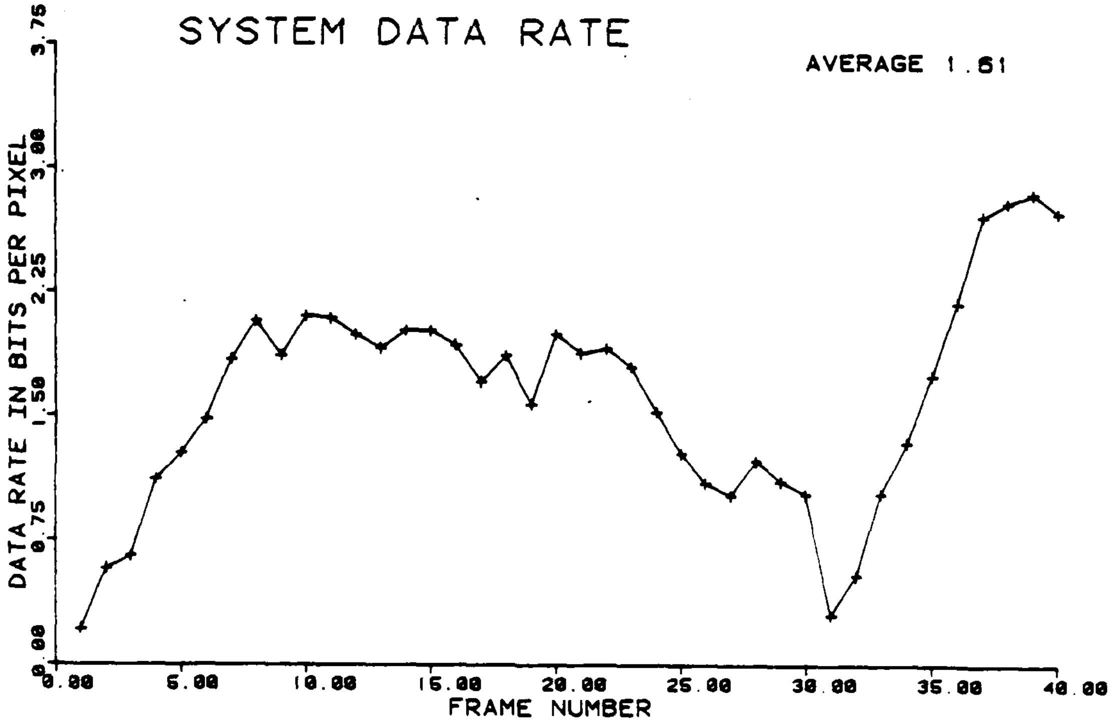


FIGURE 5.6(d) FAST PHONE. Date Rate

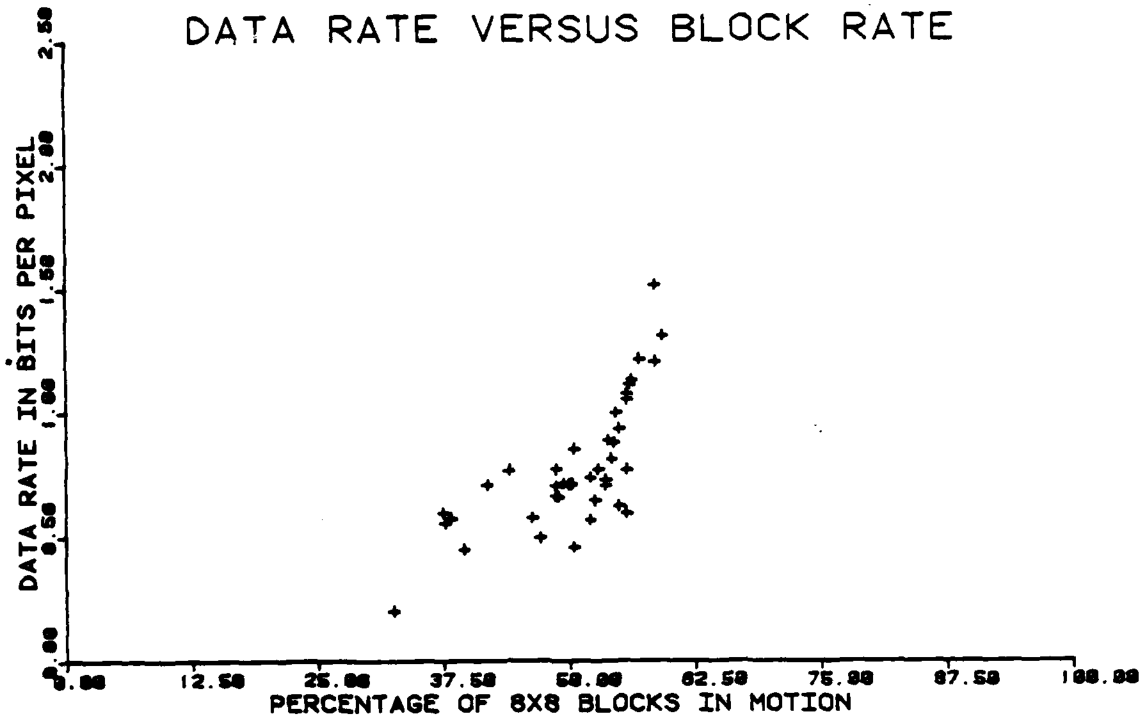


FIGURE 5.7(a) SLOW PHONE 1. Data Rate to Block Rate

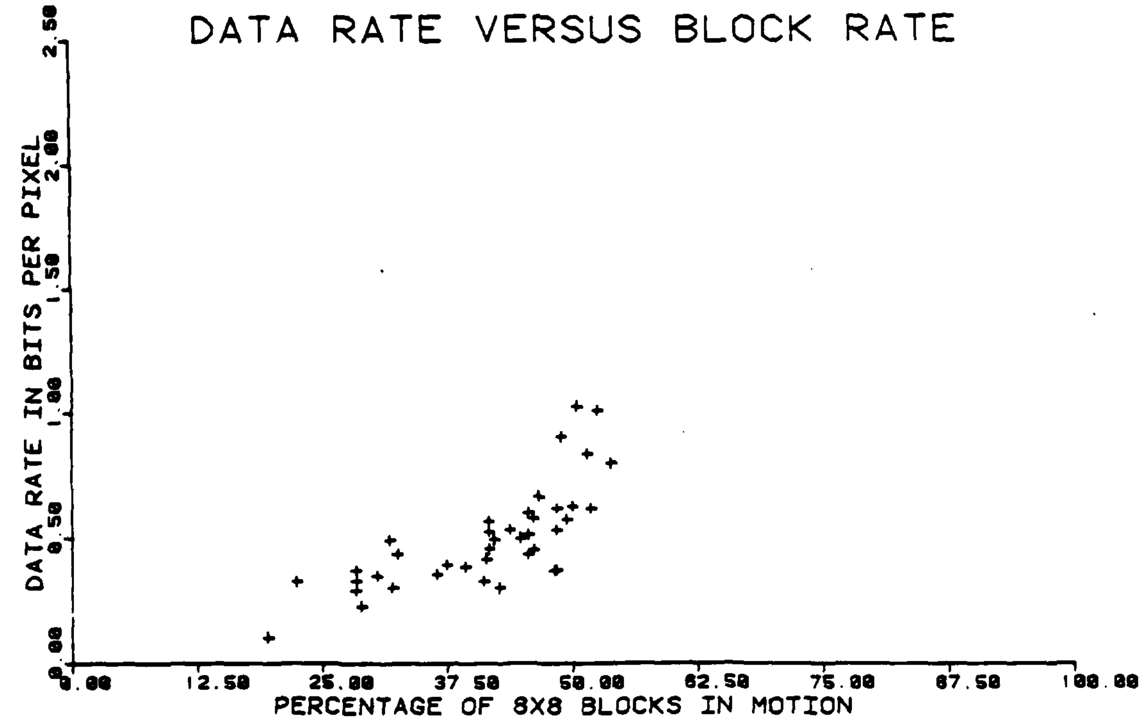


FIGURE 5.7(b) SLOW PHONE 2. Data Rate to Block Rate

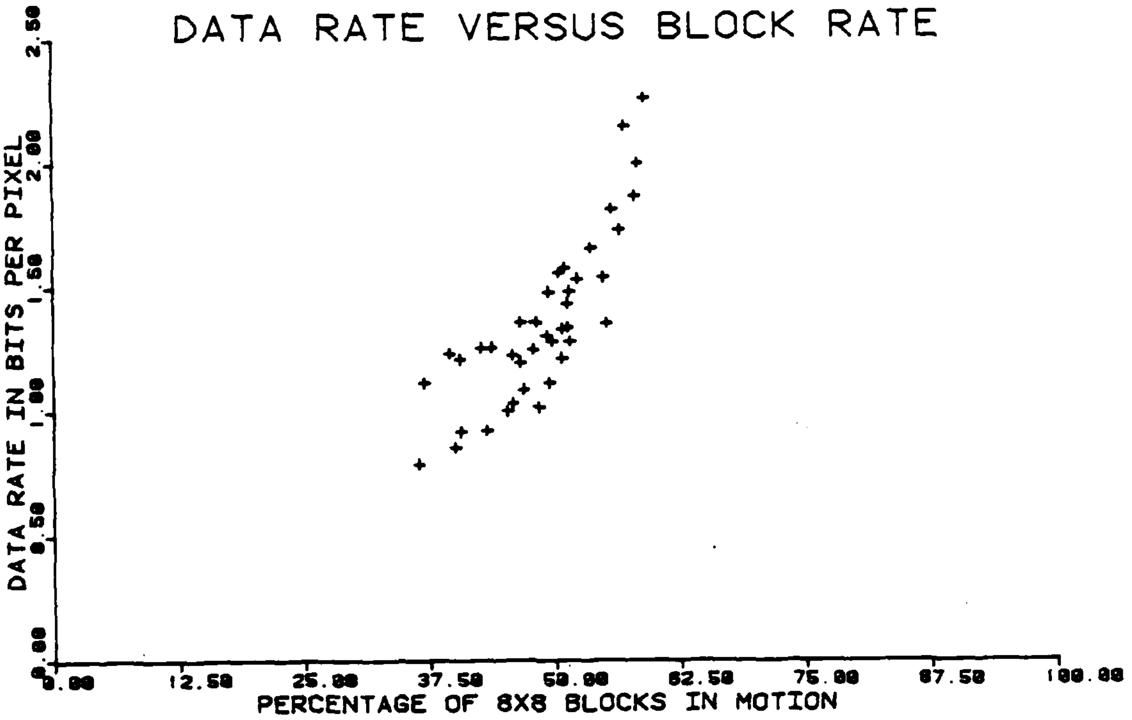


FIGURE 5.7(c) SLOW PHONE 3. Data Rate to Block Rate

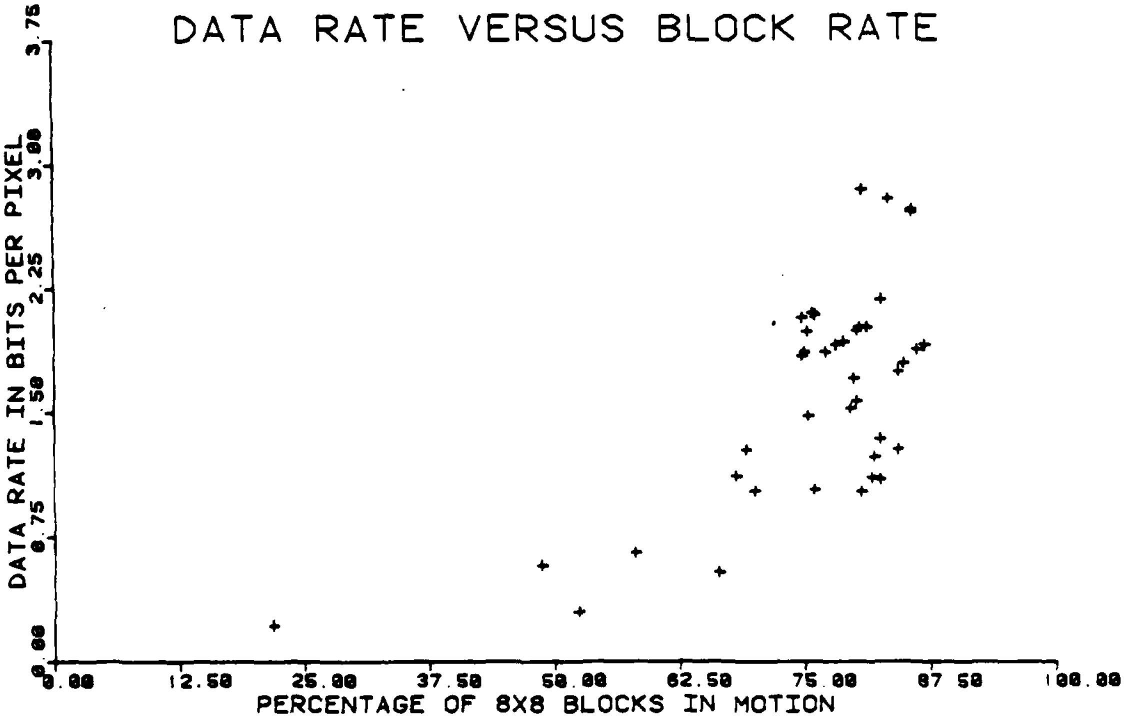


FIGURE 5.7(d) FAST PHONE. Data Rate to Block Rate

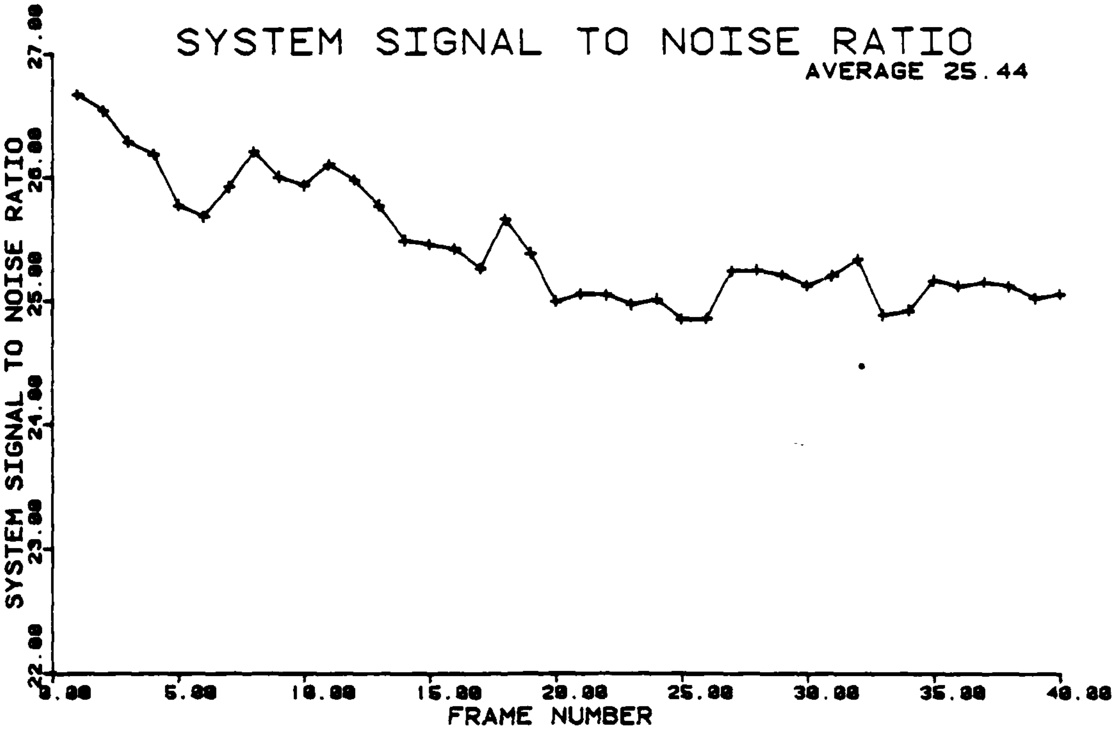


FIGURE 5.8(a) SLOW PHONE 1. System Signal To Noise Ratio

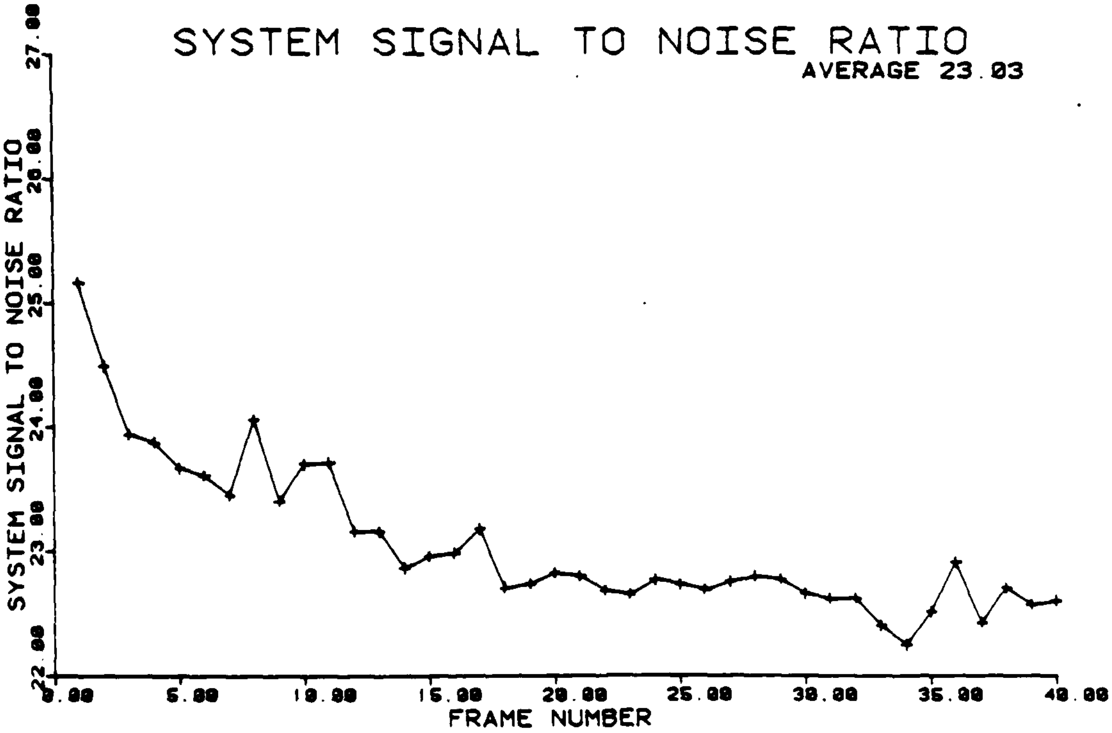


FIGURE 5.8(b) SLOW PHONE 2. System Signal To Noise Ratio

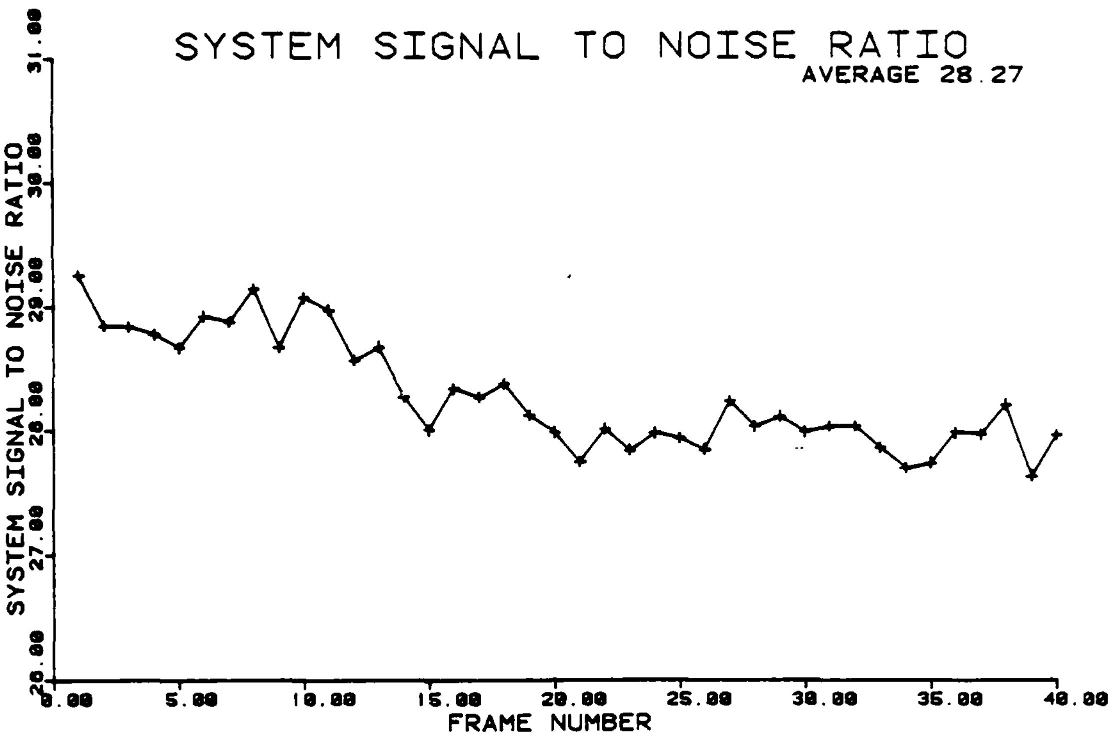


FIGURE 5.8(c) SLOW PHONE 3. System Signal To Noise Ratio

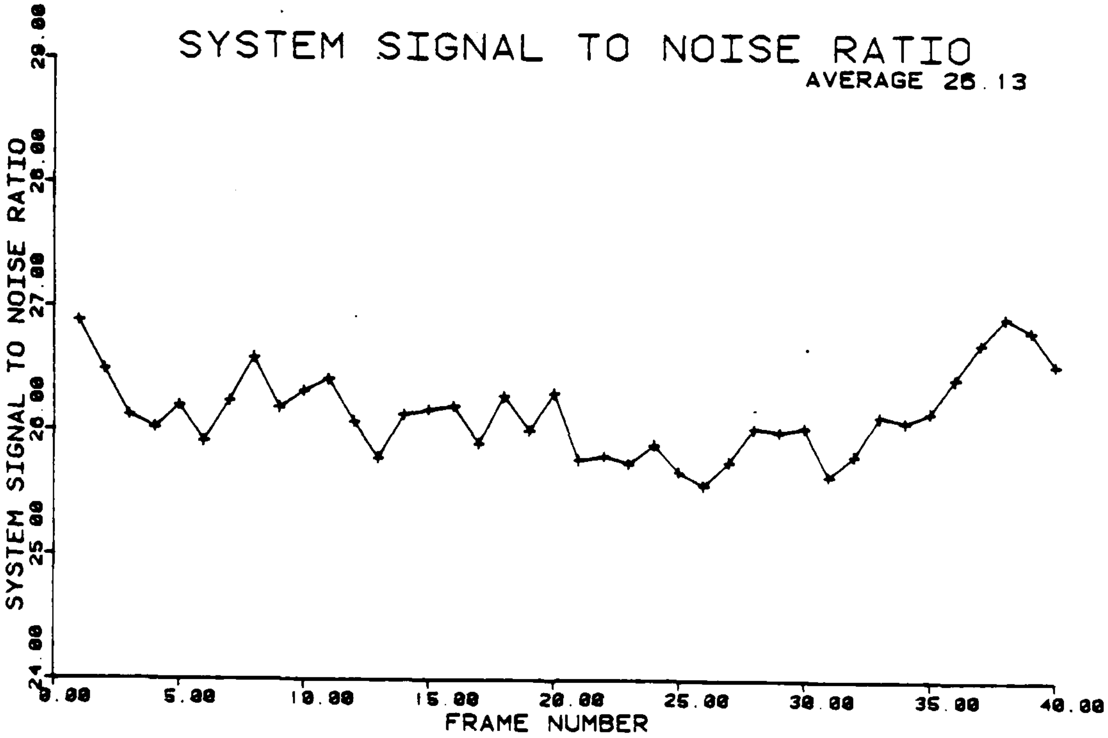


FIGURE 5.8(d) FAST PHONE. System Signal To Noise Ratio

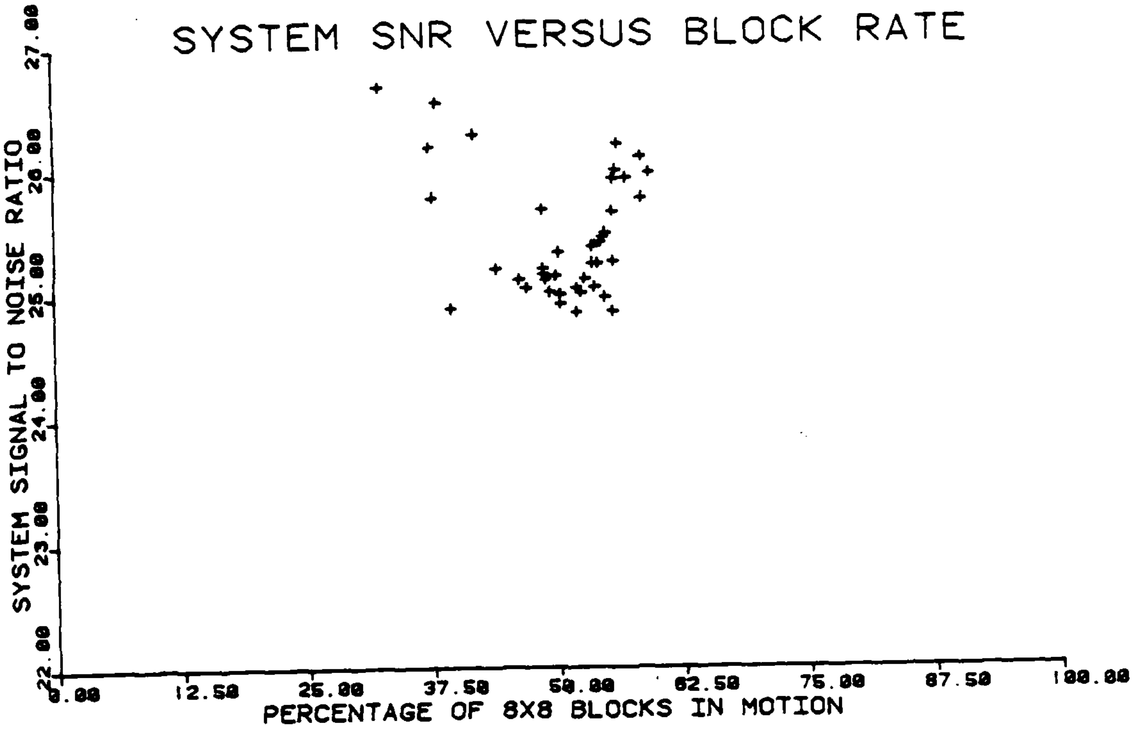


FIGURE 5.9(a) SLOW PHONE 1. System SNR to Block Rate

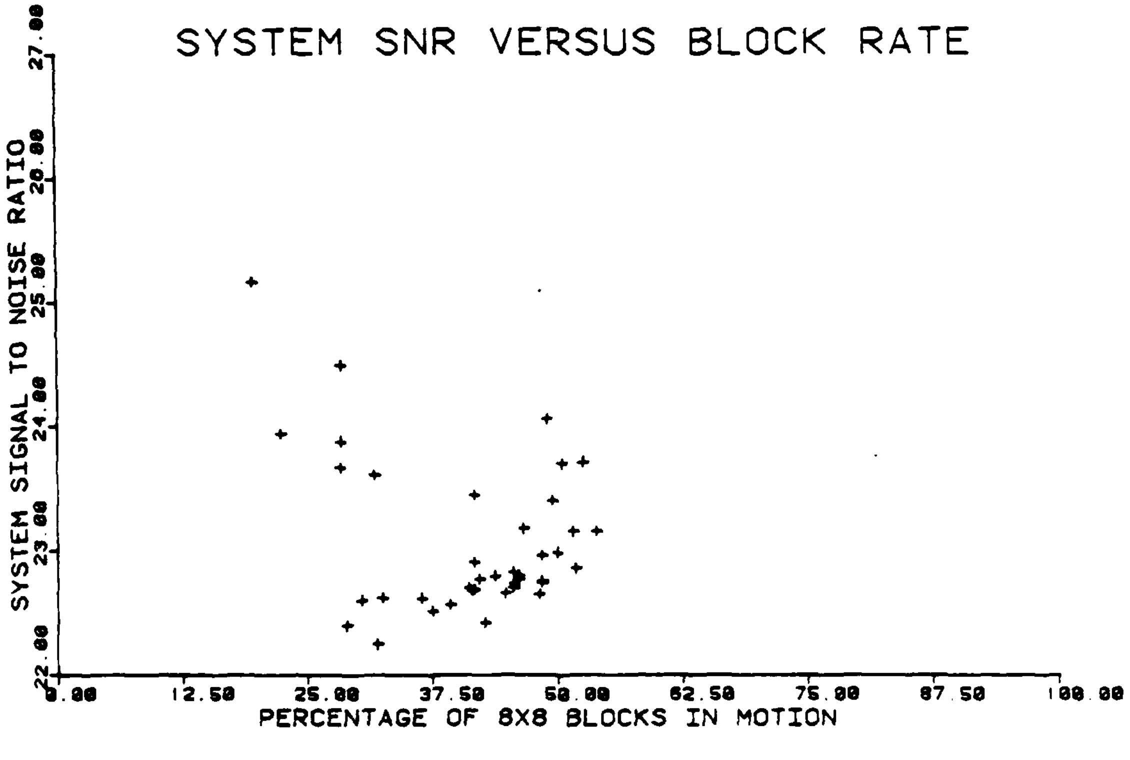


FIGURE 5.9(b) SLOW PHONE 2. System SNR to Block Rate

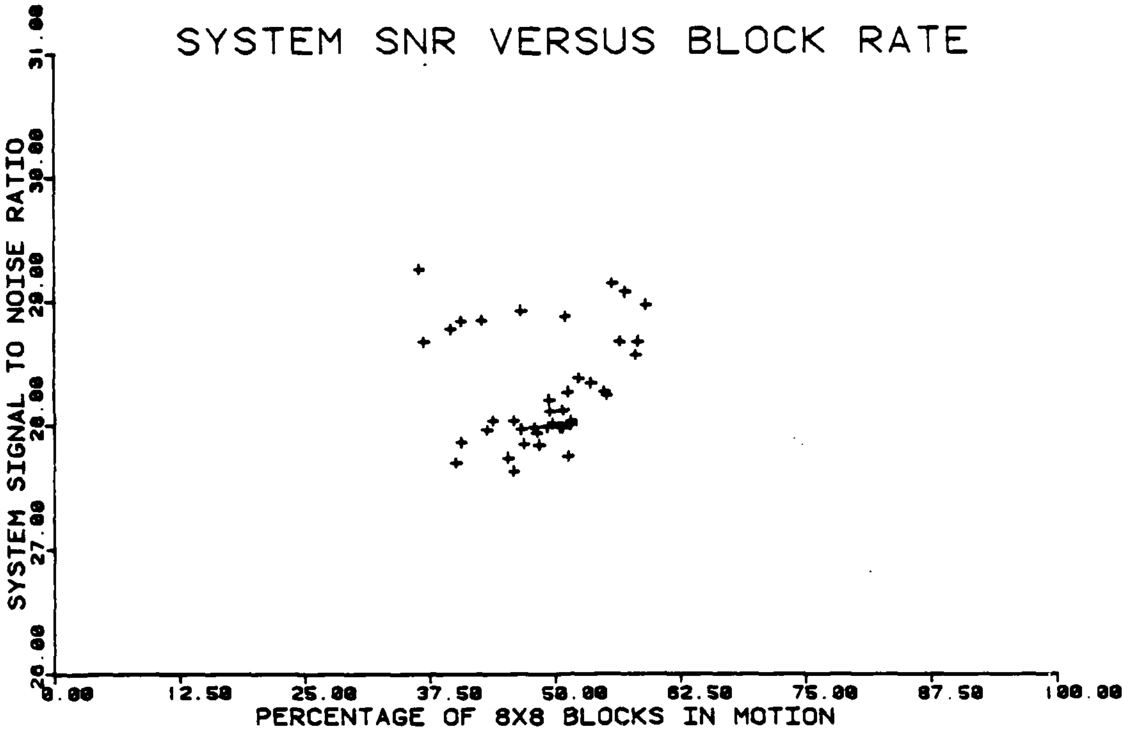


FIGURE 5.9(c) SLOW PHONE 3. System SNR to Block Rate

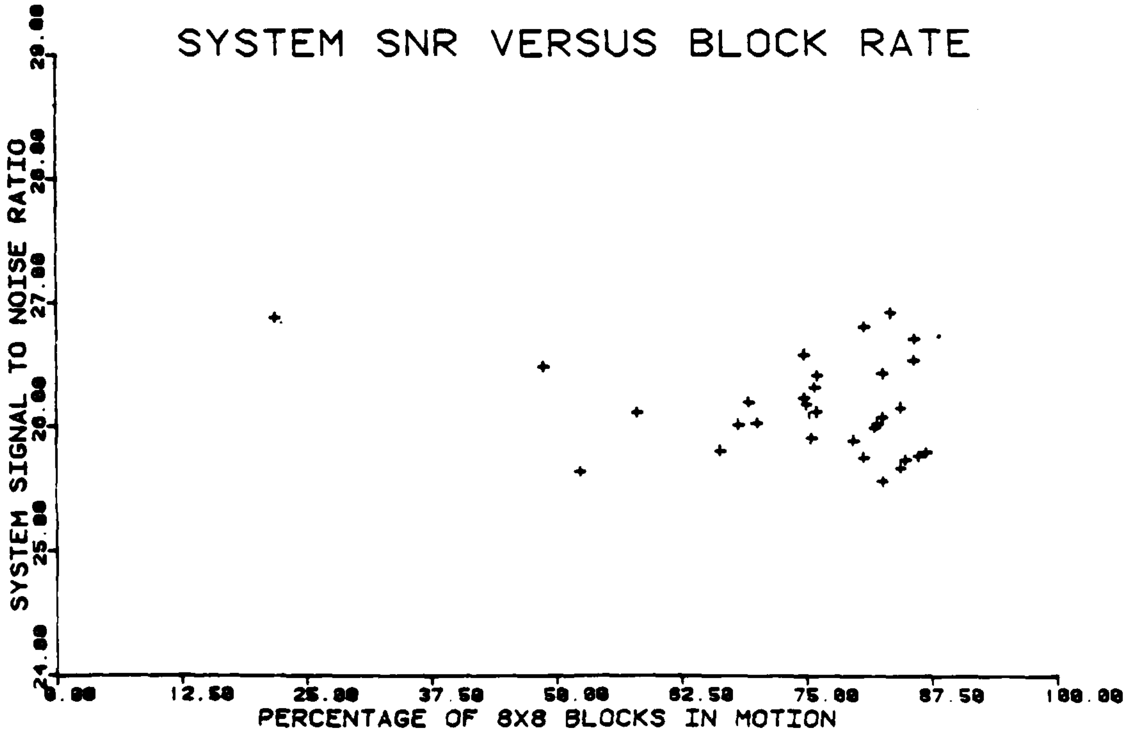


FIGURE 5.9(d) FAST PHONE. System SNR to Block Rate

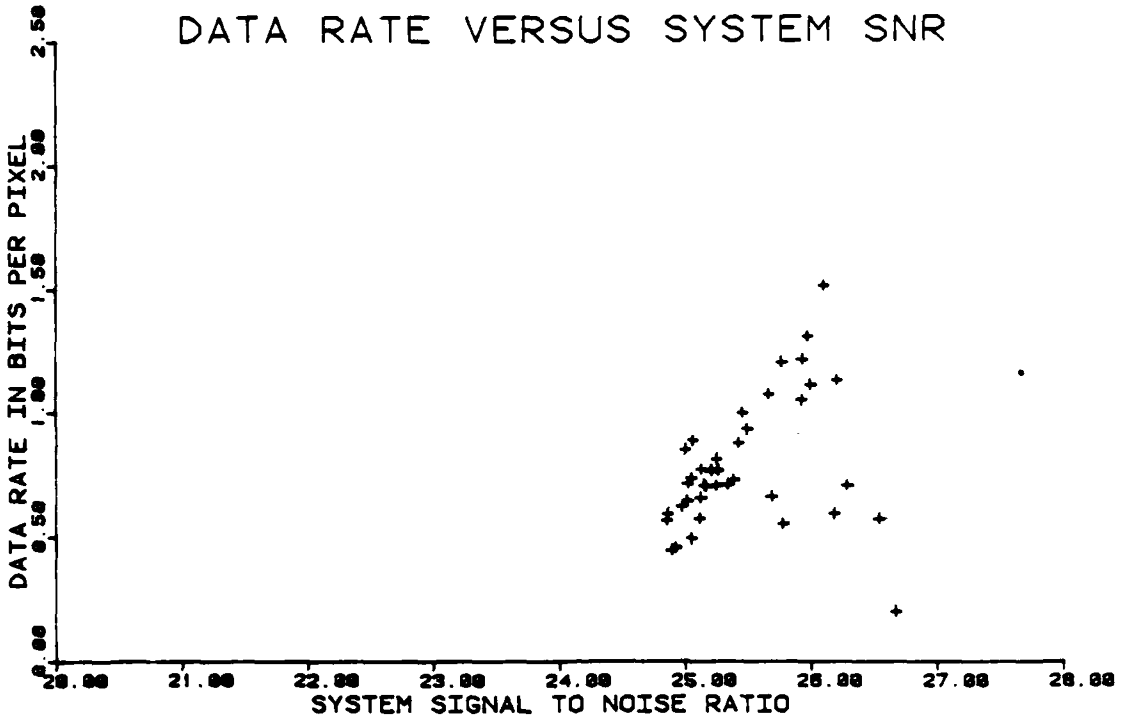


FIGURE 5.10(a) SLOW PHONE 1. System SNR to Data Rate

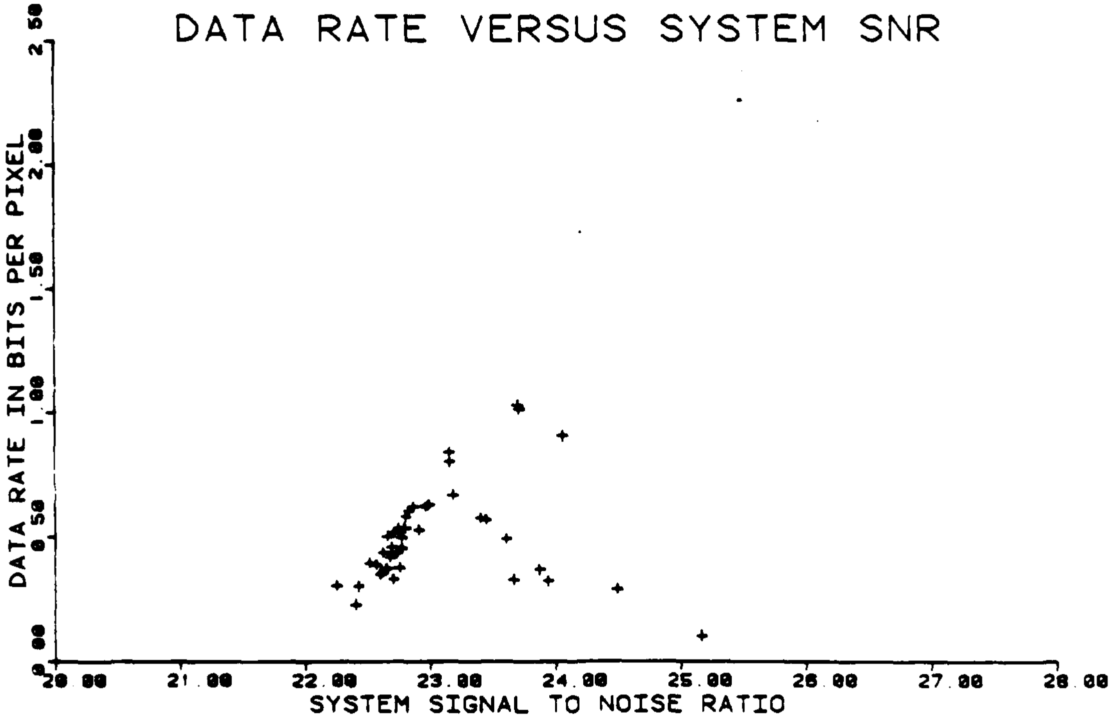


FIGURE 5.10(b) SLOW PHONE 2. System SNR to Data Rate

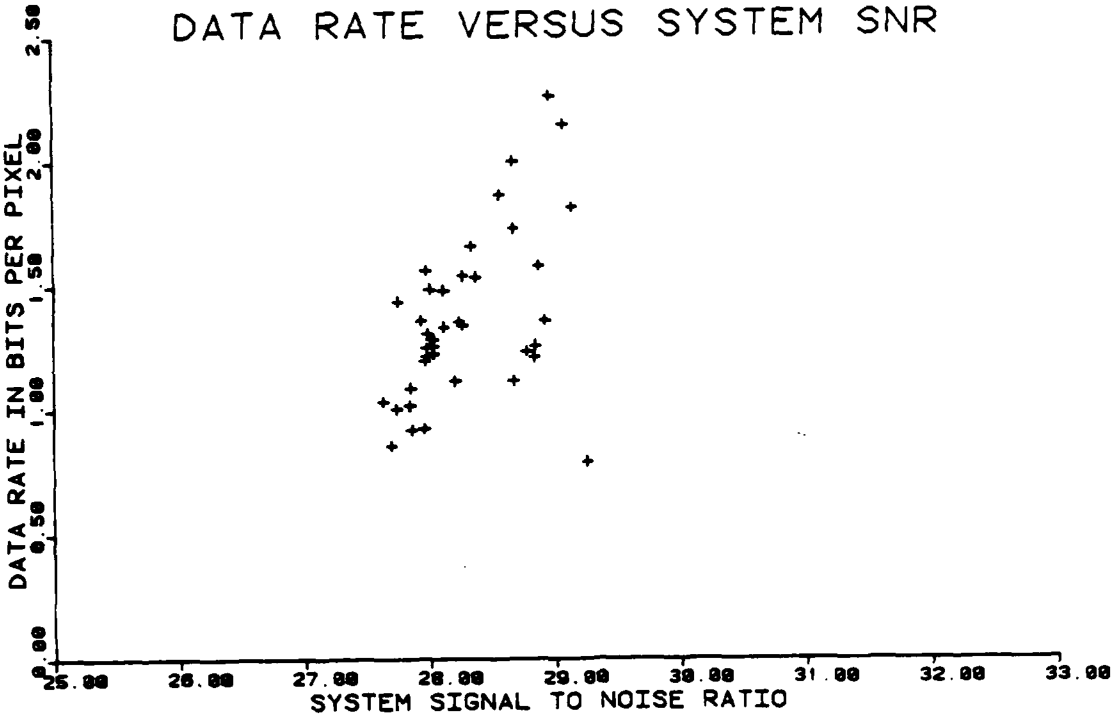


FIGURE 5.10(c) SLOW PHONE 3. System SNR to Data Rate

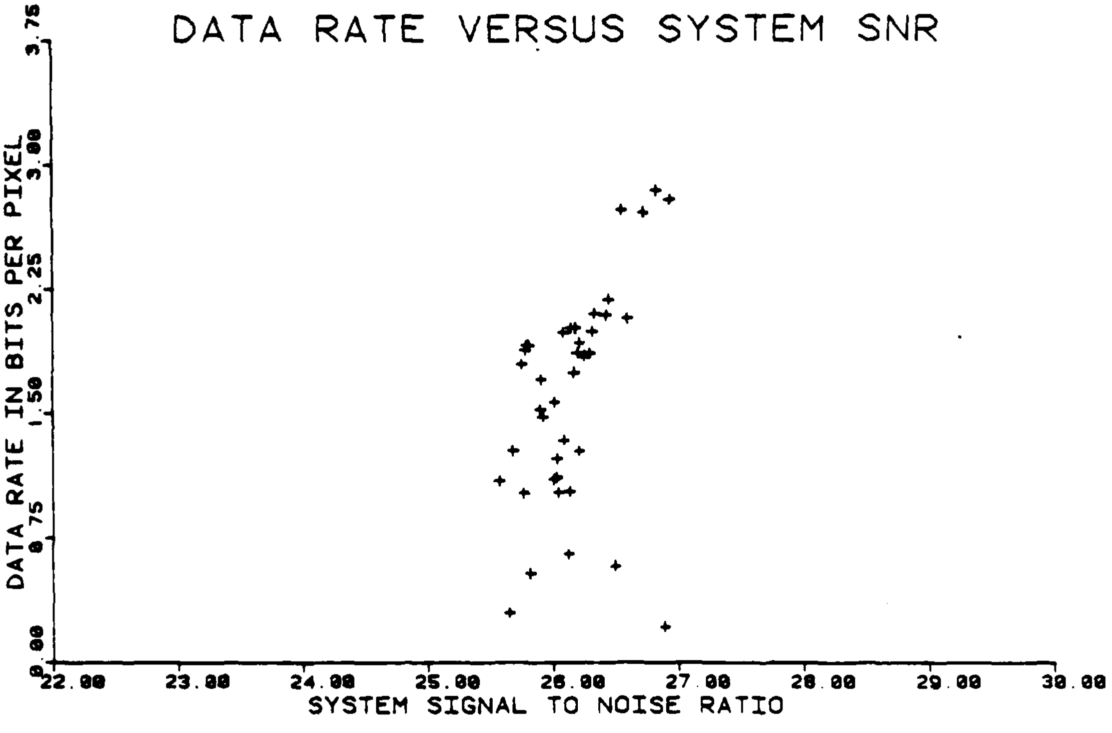


FIGURE 5.10(d) FAST PHONE. System SNR to Data Rate

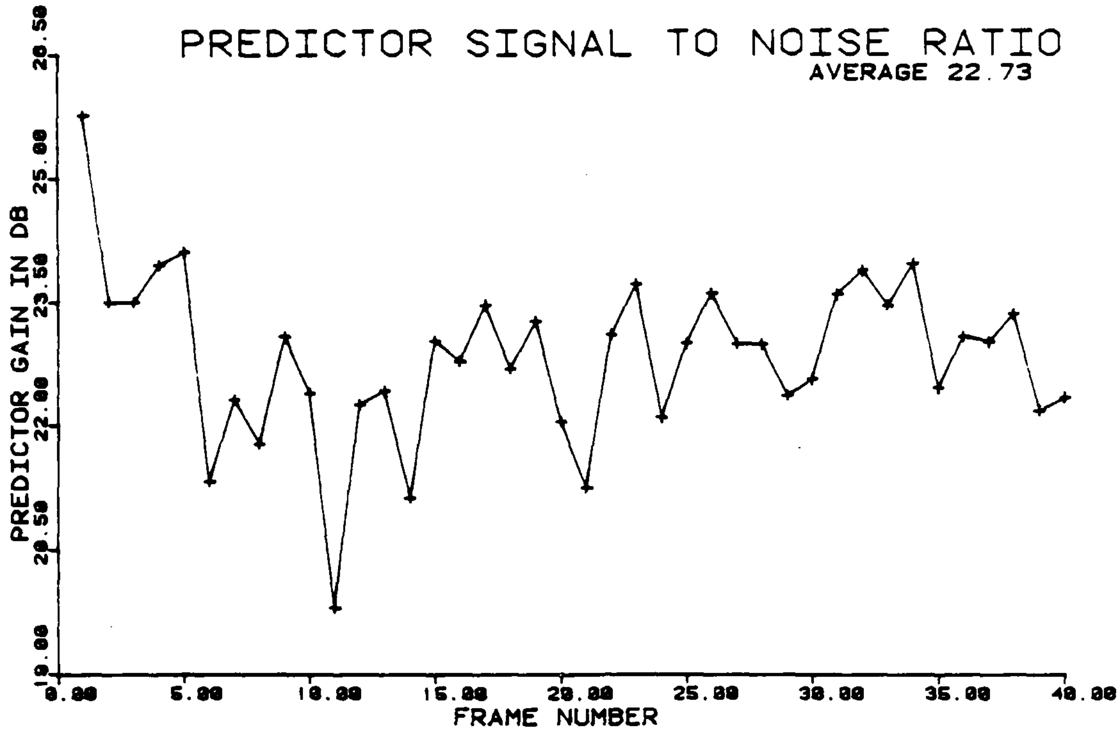


FIGURE 5.11(a) SLOW PHONE 1. Predictor Gain

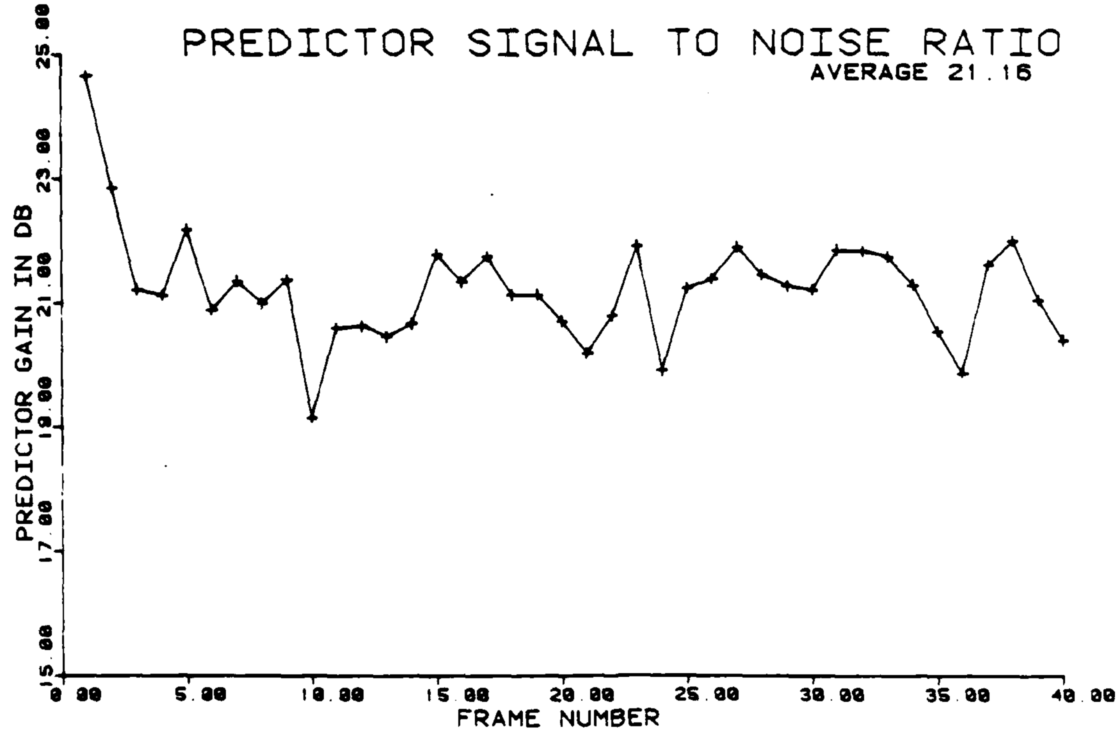


FIGURE 5.11(b) SLOW PHONE 2. Predictor Gain

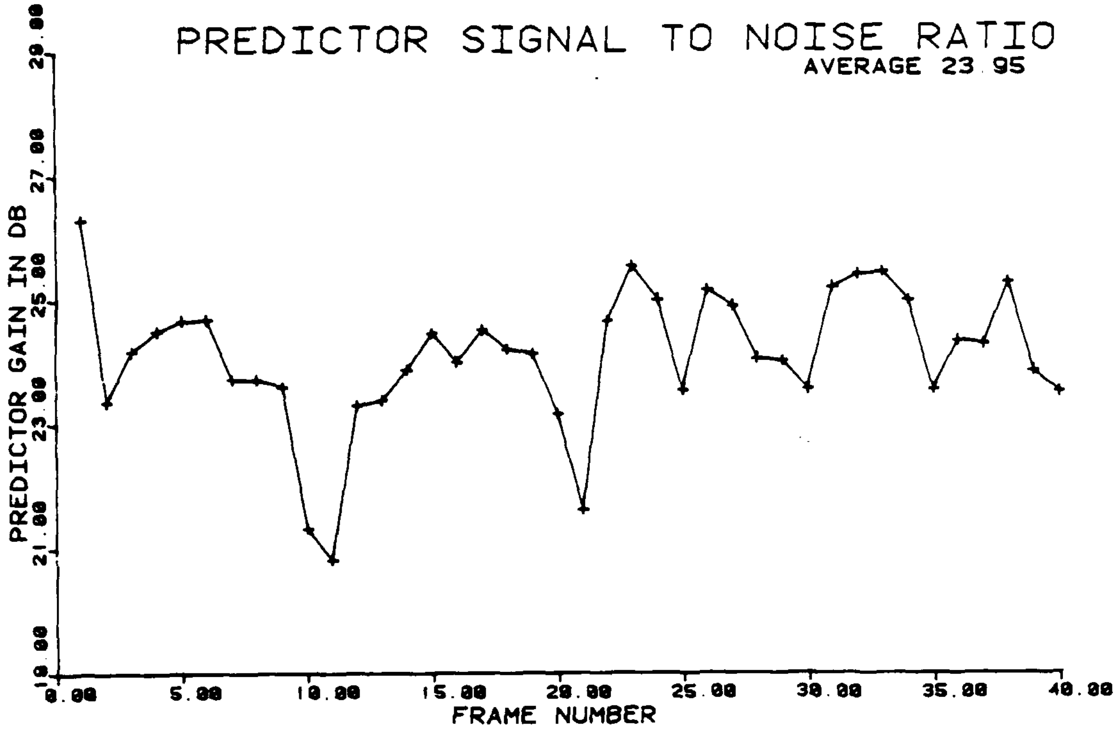


FIGURE 5.11(c) SLOW PHONE 3. Predictor Gain

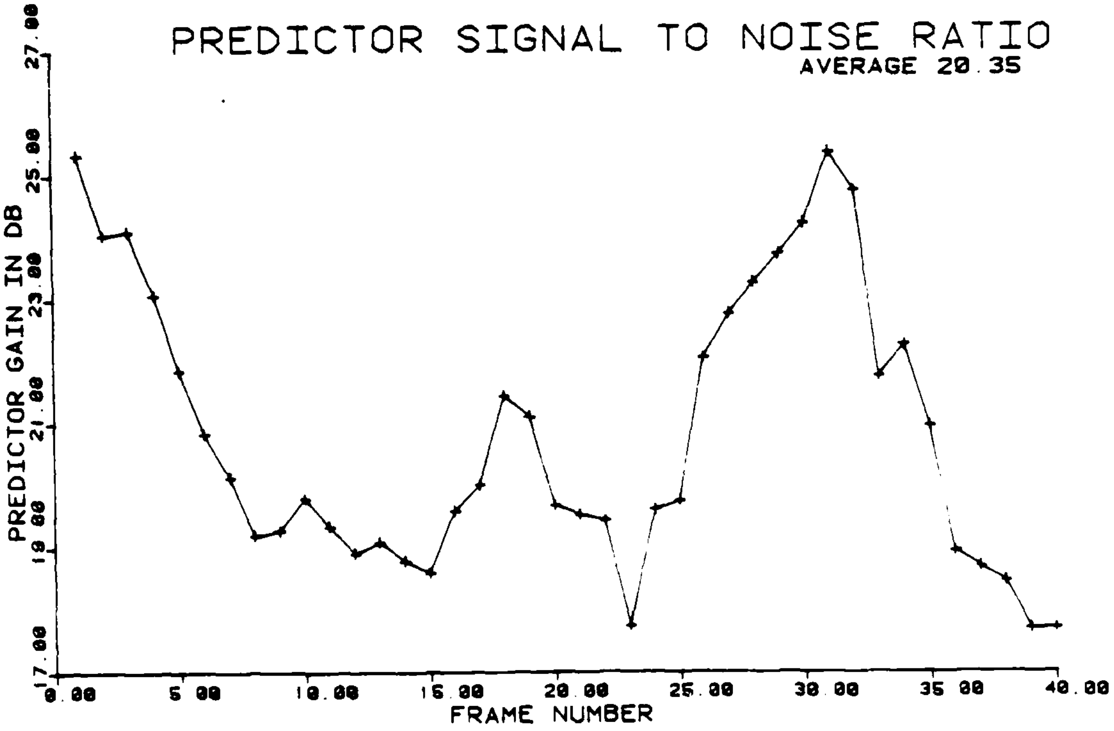


FIGURE 5.11(d) FAST PHONE. Predictor Gain

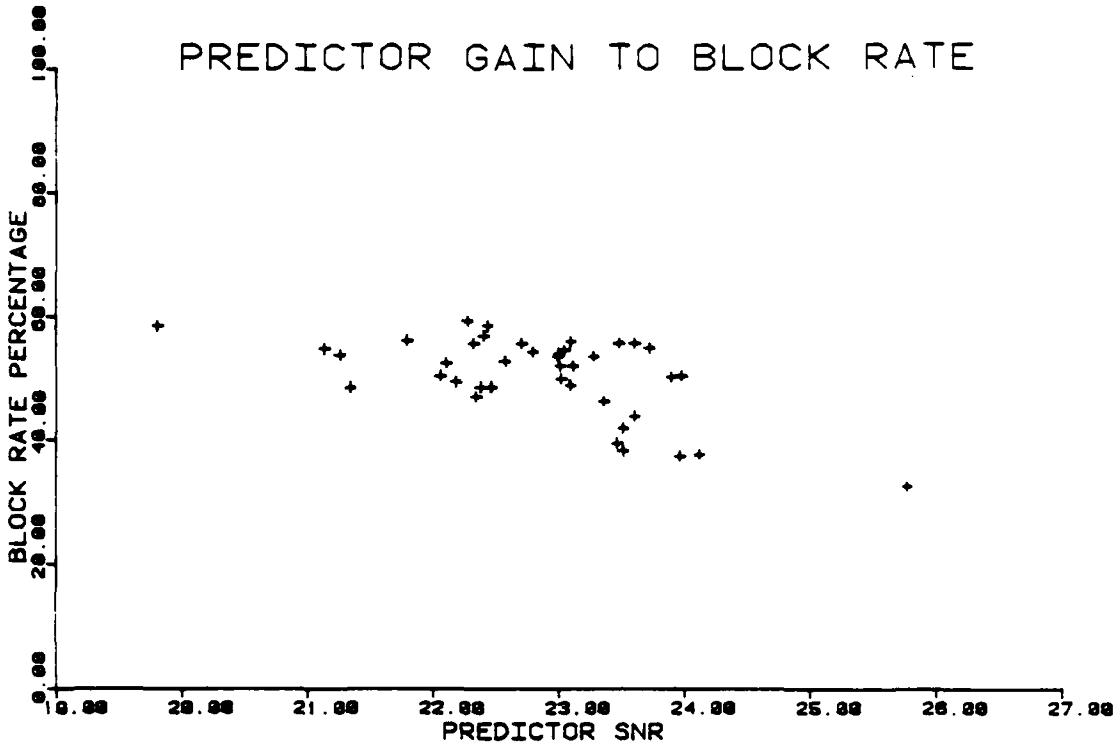


FIGURE 5.12(a) SLOW PHONE 1. Block Rate to Predictor Gain

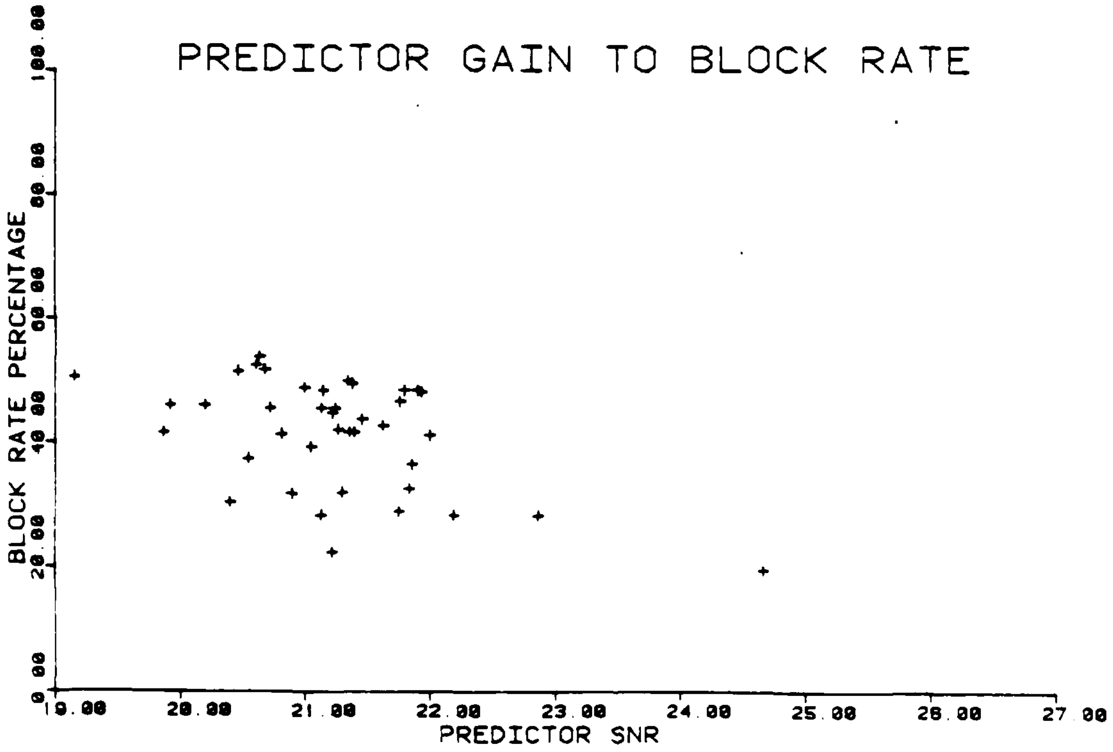


FIGURE 5.12(b) SLOW PHONE 2. Block Rate to Predictor Gain

[37] Habibi, Ali, 'Survey of Adaptive Image Coding Techniques,' IEEE Transactions on Communications, Volume COM-25, Number 11, November 1977, pp. 1275-1284.

[38] Habibi, Ali, 'An Adaptive Strategy for Hybrid Image Coding,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1736-1740.

[39] Hall, Ernest L., Tio, James B.K., McPherson, Charles A., and Sadjadi, Firooz A., 'Measuring Curved Surfaces for Robot Vision,' IEEE Computer, Volume 15, Number 12, December 1982, pp. 42-54.

[40] Haskell, Barry G., Mounts, Frank W., and Candy, James C., 'Interframe Coding of Videotelephone Pictures,' Proceedings of the IEEE, Volume 60, Number 7, July 1972, pp. 792-800.

[41] Haskell, Barry G., Gordon, Pat L., Schmidt, Robert L., and Scattaglia, James V., 'Interframe Coding of 525-Line Monochrome Television at 1.5 Mbits/s,' IEEE Transactions on Communications, Volume COM-25, Number 11, November 1977, pp. 1339-1348.

[42] Healy, Donald J., Mitchell, O.R., 'Digital Video Bandwidth Compression Using Block Truncation Coding,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1809-1817.

[43] Huang, T.S., Picture Processing and Digital Filtering, Springer-Verlag, 1975.

[44] Jacobus, Charles J., Chien, Robert T. and Selander, John M., 'Motion Detection and Analysis of Matching Graphs of Intermediate-Level Primitives,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 495-510.

[45] Jain, Anil K., and Angel, Edward, 'Image Restoration, Modelling, and Reduction of Dimensionality,' IEEE Transactions on Computers, Volume C-23, Number 5, May 1974, pp. 470-476.

[46] Jain, A.K., 'Advances in Mathematical Models for Image Processing,' Proceedings of the IEEE, Volume 69, Number 5, May 1981, pp. 502-528.

[47] Jain, Jaswant R., 'Displacement Measurement and its Application in Interframe Image Coding,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1799-1808.

[48] Jain, R., 'Extraction of Motion Information from Peripheral Processes,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-3, Number 5, September 1981, pp. 489-503.

[49] Jain, Ramesh, 'Dynamic Scene Analysis Using Pixel-Based Processes,' IEEE Computer, Volume 14, Number 8, August 1981, pp. 12-18.

[50] Jain, Ramesh, and Haynes, Susan, 'Imprecision in Computer Vision,' IEEE Computer, Volume 15, Number 8, August 1982, pp. 39-48.

[51] Jarvis, John F., 'Visual Inspection Automation,' IEEE Computer, Volume 13, Number 5, May 1980, pp. 32-38.

[52] Jarvis, R.A., 'A Computer Vision and Robotics Laboratory,' IEEE Computer, Volume 15, Number 6, June 1982, pp. 8-22.

[53] Jarvis, John F., 'Research Directions in Industrial Machine Vision: A Workshop Summary,' IEEE Computer, Volume 15, Number 12, December 1982, pp. 55-61.

[54] Jones, R.A., 'Adaptive Hybrid Picture Coding,' SPIE, Volume 87, Advances in Images Processing Techniques, 1976, pp. 247-255.

[18] Cooper, G.R., and McGillem, C.D., Probabilistic Methods of Signal and System Analysis, Holt, Rinehart, and Winston Inc., 1971.

[19] Daut, D.G., Fries, R.W., and Modestino, J.W., 'Two-Dimensional DPCM Image Coding Based on an Assumed Stochastic Image Model,' IEEE Transactions on Communications, Volume COM-29, Number 9, September 1981, pp. 1365-1374.

[20] Drake, Alvin W., Fundamentals of Applied Probability Theory, McGraw Hill, New York, 1967.

[21] Draper, N.R. and Smith, H., Applied Regression Analysis, John Wiley and Sons, New York, 1966.

[22] Dudewicz, Edward J., Introduction to Statistics and Probability, Holt, Rinehart and Winston, New York, 1976.

[23] Dukhovich, Isaac J., and O'Neal, J.B. Jr., 'A Three-Dimensional Spatial Non-Linear Predictor for Television,' IEEE Transactions on Communications, Volume COM-26, Number 5, May 1978, pp. 578-583.

[24] Ferrie, Frank P., Levine, Martin D., and Zucker, Steven W., 'Cell Tracking: A Modeling and Minimization Approach,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-4, Number 3, May 1982, pp. 277-291.

[25] Forgus, R.H., and Melamed, L.E., Perception: A Cognitive Stage Approach, McGraw-Hill Book Company, 1976.

[26] FU, King-sun, 'Pattern Recognition for Automatic Visual Inspection,' IEEE Computer, Volume 15, Number 12, December 1982, pp. 34-40.

[27] Gallager, R.G., Information Theory and Reliable Communication, John Wiley and Sons Inc., 1968.

[28] Gilbert, Alton L., Giles, Michael K., Flachs, Gerald M., Rogers, Robert B., and U, Yee Hsun, 'A Real-Time Video Tracking System,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 1, January 1980, pp. 47-56.

[29] Gilbert, Alton L., 'Video Data Conversion and Real-Time Tracking,' IEEE Computer, Volume 14, Number 8, August 1981, pp. 50-56.

[30] Giorda, F. and Racciu, A., 'Bandwidth Reduction of Video Signals via Shift Vector Transmission,', IEEE Transactions on Communications, Volume COM-23, Number 9, September 1975, pp. 1002-1004.

[31] Gonzalez, R.C., and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Company Inc., 1979.

[32] Gonzalez, Rafael C. and Safavakhsh, Reza, 'Computer Vision Techniques for Industrial Applications and Robot Control,' IEEE Computer, Volume 15, Number 12, December 1982, pp. 17-32.

[33] Goyal, Shri K., and O'Neal, J.B. Jr., 'Entropy Coded Differential Pulse-Code Modulation Systems for Television,' IEEE Transactions on Communications, Volume COM-23, Number 6, June 1975, pp. 660-666.

[34] Granrath, Douglas J., 'The Role of Human Visual Models in Image Processing,' Proceedings of the IEEE, Volume 69, Number 5, May 1981, pp. 552-561.

[35] Habibi, Ali, and Wintz, Paul A., 'Image Coding by Linear Transformation and Block Quantization,' IEEE Transactions on Communications, Volume COM-19, Number 1, February 1971, pp. 50-62.

[36] Habibi, Ali, 'Hybrid Coding of Pictorial Data,' IEEE Transactions on Communications, Volume COM-22, Number 5, May 1974, pp. 614-624.

BIBLIOGRAPHY

[1] Agin, G.J., 'Computer Vision Systems for Industrial Inspection and Assembly,' IEEE Computer, Volume 13, Number 5, May 1980, pp. 11-20.

[2] Alexander, Peter, 'Array Processors in Medical Imaging,' IEEE Computer, Volume 16, Number 6, June 1983, pp. 17-30.

[3] Anderson, B.D.O., and Moore, J.B., Optimal Filtering, Prentice-Hall Inc., 1979.

[4] Andrews, H.C., and Hunt, B.R., Digital Image Restoration, Prentice-Hall Inc., 1977.

[5] Ayala, I.L., Orton, D.A., Larson, J.B., and Elliot, D.F., 'Moving Target Tracking Using Symbolic Registration,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-4, Number 5, September 1982, pp. 515-520.

[6] Berger, T., Rate Distortion Theory, Prentice Hall, Englewood Cliff, New Jersey, 1971.

[7] Bowling, Carl D., and Jones, R.A., 'Displacement Estimation by Prediction Coefficient Energy Concentration,' IEEE Global Telecommunications Conference, GLOBECOM'82, Volume 1, pp. B6.4.1-B6.4.4.

[8] Bowling, Carl D., 'An Innovations Approach to Edge Detection,' Masters Thesis, Departement of Electrical Engineering, University of Arkansas, December 1980.

[9] Bozic, S.M., Digital and Kalman Filtering, E. Arnold, London, 1979.

[10] Brown, Robert Grover, Random Signal Analysis and Kalman Filtering, John Wiley and Sons, New York, 1983.

[11] Budrikis, Z.L., 'Visual Fidelity Criterion and Modeling,' Proceedings of the IEEE, Volume 60, Number 7, July 1972, pp. 771-779.

[12] Cafforio, Ciro and Rocca, Fabio, 'Methods for Measuring Small Displacements of Television Images,' IEEE Transactions on Information Theory, Volume IT-22, Number 5, September 1976, pp. 573-579.

[13] Cannon, T.M. and Hunt, B.R., 'Image Processing by Computer,' Scientific American, Volume 245, Number 4, October 1981, pp. 214-225.

[14] Castleman, K.R., Digital Image Processing, Prentice-Hall Inc., 1979.

[15] Chen, Wen-Hsiung, and Smith, C. Harrison, 'Adaptive Coding of Monochrome and Color Images,' IEEE Transactions on Communications, Volume COM-25, Number 11, November 1977, pp. 1285-1292.

[16] Clark, D.C. and Gonzalez, R.C., 'Optimal Solution of Linear Inequalities with Applications to Pattern Recognition,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-3, Number 6, November 1981, pp. 643-655.

[17] Connor, Denis J. and Limb, John O., 'Properties of Frame-Difference Signals Generated by Moving Images,' IEEE Transactions on Communications, Volume COM-22, Number 10, October 1974, pp. 1564-1575.

BIBLIOGRAPHY

The iteration methods are also unable to cop with occlusion as well as non-translation types of motion. The Prediction Coefficient method is able to partially alleviate these problems.

The results presented show a good degree of data compression, while at the same time producing high quality output. The algorithm has been tested on data similar to that which would be encountered in a videotelephone setting. A large amount of the algorithm analysis has been performed. To some degree it has been shown how various system parameters effect the quality of the output and the data rate required to achieve it. Also, the interaction among the predictor gain, the gain produced by the quantizer, and the overall system gain have been investigated. Actual output images have been presented for visual comparison of the different preset parameters. Ideas and suggestions were provided for possible future research to continue with the work that has been started here.

requirement in background and shadow areas than would be expected. In these background and shadow areas, both the signal variance and signal power are low. There may be a way to take into account both the signal variance and the signal power to improve the quantizing quality.

Finally, work is needed in the area of noise effects. A study of how noise in the input sequence effects both the prediction process and the output quality of the image sequence is needed. More importantly, how noise in the digital channel effects the operation of the algorithm, as well as the quality of the final output, needs fu-ther research. The implementation here assumed a noise free channel, but for real world applications an additive noise channel will need to be modelled.

Only a few of the many possible refinements and extensions of this one particular method for motion compensated image coding have been mentioned. The entire field of motion compensated image coding is still in its infancy, with an enormous amount of work remaining. To even attempt to list all of the possible paths for future research would be a major undertaking. The fields of research remain wide open.

5.3 CONCLUSIONS

The method of Prediction Coefficient Energy Concentration has proved to yield an improvement in motion compensated image coding when compared with previously published results. The improvements come in both the areas of algorithm complexity and in the method for determining the displacement estimation. Previous methods for displacement estimation have relied on an iterative procedure that was both time consuming and rigidly fixed to translation types of motion only. The method of Prediction Coefficient Energy Concentration has replaced the iterative estimation procedure with a two-step estimation procedure. The first step produces an estimate for the integer portion of the displacement. The second step, if needed, produces a set of predictor coefficients that are able to perform the same function as the non-integer portion of the displacement.

The algorithm does not have the problems of convergence that many of the previous methods exhibit. The replacement of the iteration procedure with the two-step prediction process relieves this problem.

Further work is required to determine the effectiveness of this procedure as well as how this setting effects the actual quality of the output. Signal to noise ratio is a term that is easily calculated, but may not have a direct relationship to the actual quality of the output image sequence. Perhaps some other figure of merit would function better.

The area of the procedure that requires the majority of the bandwidth is also the area that offers the greatest potential for further research. This is the area of error quantization. The system as presented here, assumed the error distribution to be Gaussian and attempted to minimize the sum of the absolute value of the error. This is a major oversimplification of the data that actually results. It is known that the distribution is not Gaussian, but estimates of the actual distribution are unknown. Better estimates of the actual error distribution are required. Perhaps one solution would be to define a set of possible distributions and determine which of these distributions the current residual data best fits. It is expected that the errors that arise from blocks located at the moving/non-moving interface would differ substantially from those blocks that tend to have a constant translation. Yet another distribution would be required for areas where background is uncovered. If it could be determined from which area a block originated, an estimate for that local error distribution might be generated.

Another problem in the area of quantization that merits further research, is that of possible data dependent thresholding or quantization. It has already been stated that the eye will allow a higher degree of degradation for portions of the image that are undergoing translation than it will for non-moving portions. It also seems that the human visual system is very critical about small errors when they occur in areas of the image with small signal variance. This problem was partially addressed in that the signal to noise ratio was defined as ten log the ratio of the signal variance to the error variance, as opposed to ten log the ratio of the signal power to the error power as is normally used. This forced a better residual update in the areas of the image with small signal variance, such as the face and hand areas of the images presented. It also forced a higher update alleviated with the help of a low pass filter on the output stage of the motion compensation system to smooth the output.

5.2 RECOMMENDATIONS FOR FUTURE RESEARCH

This work has only scratched the surface as far as motion compensated image coding is concerned. The groundwork material has been set and a path started, but many more questions have been asked than have been answered. Much of the algorithm analysis and implementation work yet remains.

One area of the algorithm analysis that needs to be studied in more detail, is the area of optimal threshold and parameter selection. As pointed out in the last section, effort is needed to determine optimal values of the thresholds and how the choice of one effects the output, as well as the action of some of the other parameters. Ideally, some of the parameters should be tied to the quality of the image data itself. It will prove unproductive to try to generate an output image sequence with a signal to noise ratio greater than that of the input sequence. Along the same lines, the difference in quality of display devices can also contribute to the setting of various system parameters. That is, the dynamic range and distortion of the display device needs to be considered when the image quality requirements are defined. In general, the choice for the threshold for determining if the displacement is integer or non-integer as well as the choice for the threshold for determining if sub-block processing is required, requires some more effort. These two thresholds are not unrelated. For example, if the error threshold is increased for the integer/non-integer test, allowing more blocks to pass as integer displacement, fewer will even reach the portion of the algorithm that tests for sub-block processing. Some work needs to be performed to determine what the optimal threshold settings are, how they are related to the actual image data, and also how they are related to one another.

One of the other preset system parameters that warrants further research, is the value for the output signal to noise ratio. As was noted in one of the previous chapters, when a fixed length buffer is used in the implementation, the value for this required signal to noise ratio of the output could be used as a buffer regulating parameter.

5.1 REMAINING PROBLEMS

The overall results of this method for motion compensated image coding have proved promising. A fairly high quality output has been achieved at a nominal overall data rate. Still, for video-telephone applications the data rate remains too high. There remains a number of problems with both the algorithm itself and its implementation.

Perhaps the biggest problem, other than the required bandwidth, for a video-telephone implementation, is that of computational speed. It is, and would be, impractical to try to implement the algorithm in anything other than a parallel architecture hardware device. At various points throughout the different chapters, sections of the algorithm that lend themselves to parallel implementation were noted. Real time processing of video images is a very calculation intensive undertaking that interfaces well with parallel processing techniques. The problem of processing time may be negligible when a real time hardware parallel processor is used.

Another problem of the computer simulation was that of the choice of the various thresholds and other system parameters. No attempt was made to optimize all of these parameters. Some of the thresholds and parameters were chosen to estimate the noise introduced into the system by the imaging apparatus, while others, such as the predictor coefficient coding length, were chosen by a mathematical model. The image quaiity and data rate requirements are opposing ends. That is, minimizing the data rate tends to decrease the image quality, as does increasing the image quality require an increase in data rate. Both of these are very dependent upon the settings of the various thresholds.

One of the problems that remains in the output image can only be seen when the images are viewed in sequence as they were intended to be. This problem comes about because of the ability of the human eye and visual system to detect very minute changes in gray level, if it has some geometric structure. This can be seen to occur in the image sequence at lower data rates when the outlines of the sub-blocks of the images become visible. Because of the straight line geometric nature of the block boundaries, they appear more noticeable than one would expect for the size of the actual error. Perhaps this could be partially

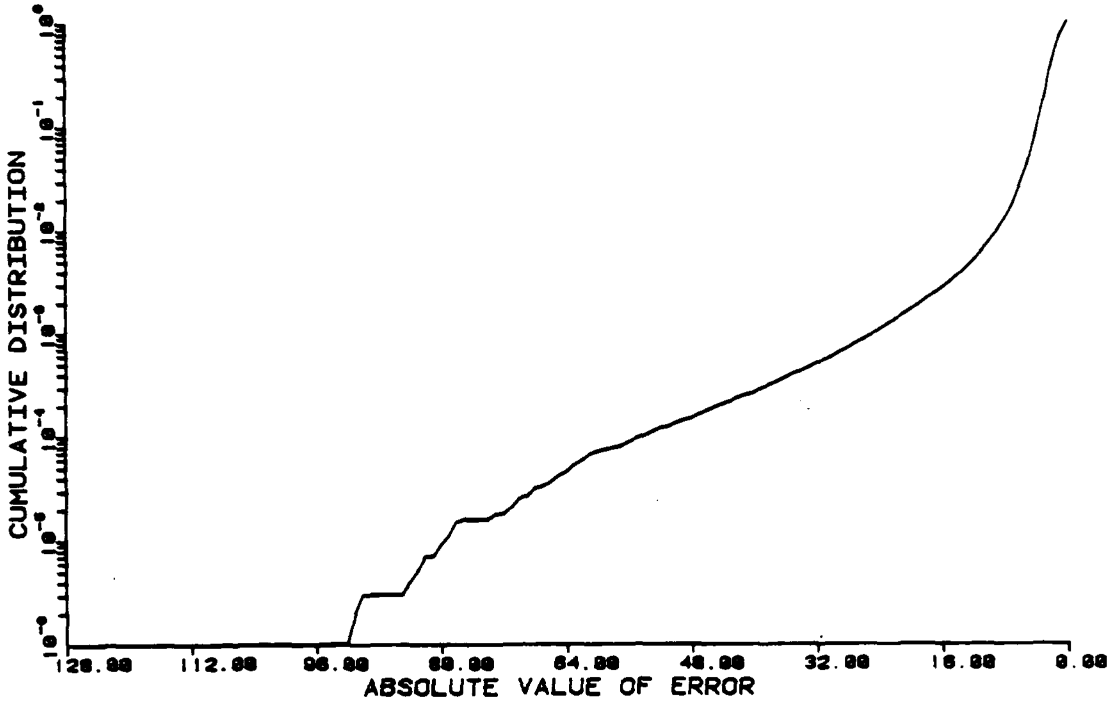


FIGURE 5.15(c) SLOW PHONE 3. Log Cumulative Error Distribution

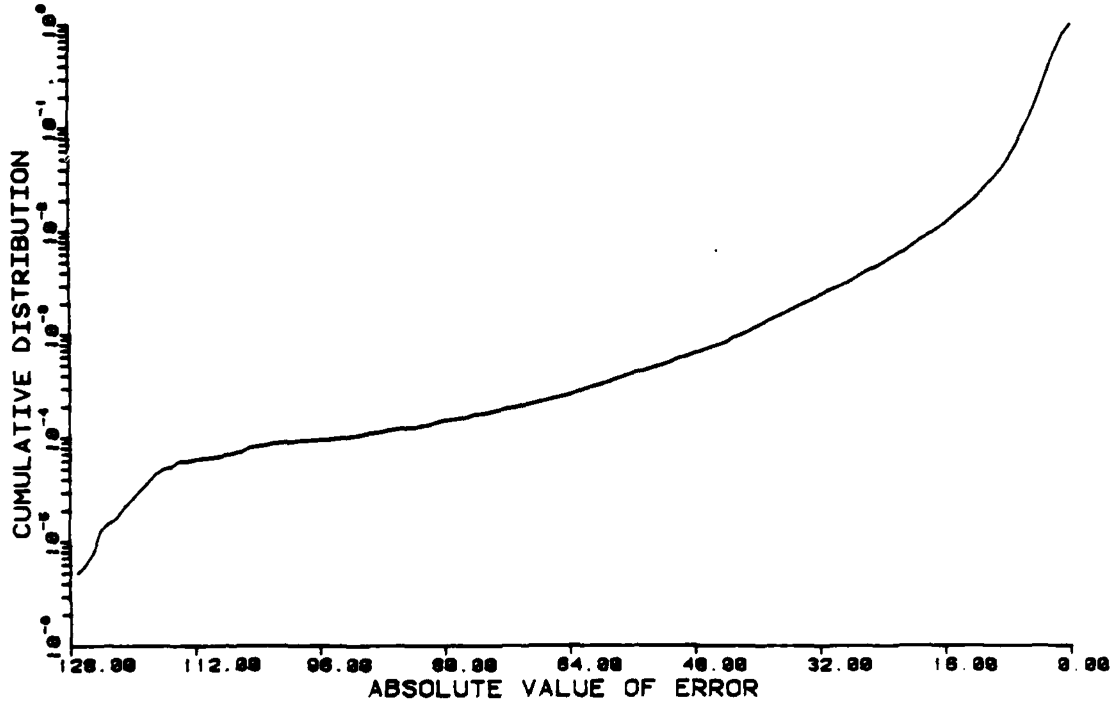


FIGURE 5.15(d) FAST PHONE. Log Cumulative Error Distribution

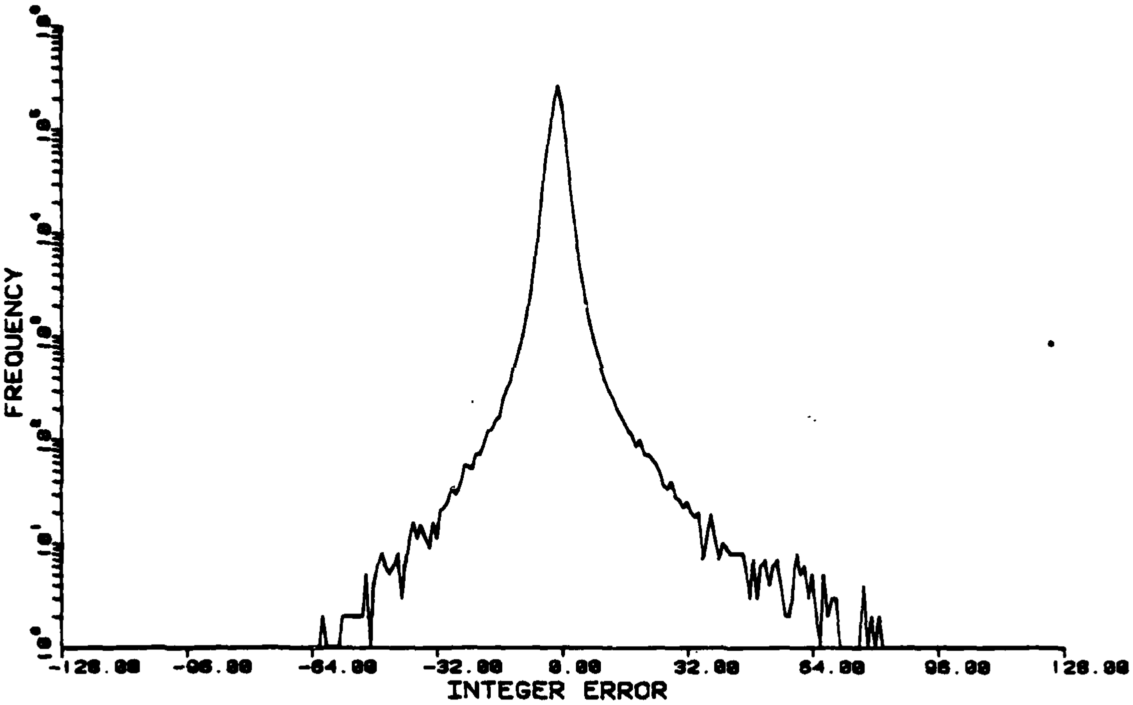


FIGURE 5.14(c) SLOW PHONE 3. Log of Error Distribution

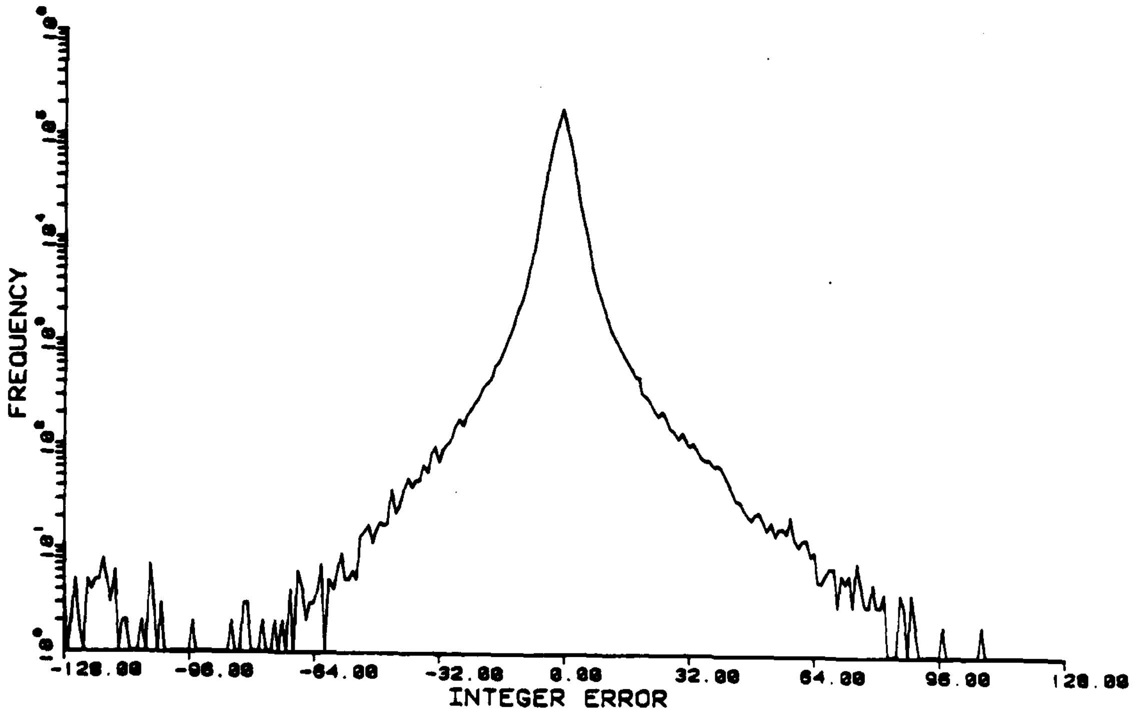


FIGURE 5.14(d) FAST PHONE. Log of Error Distribution

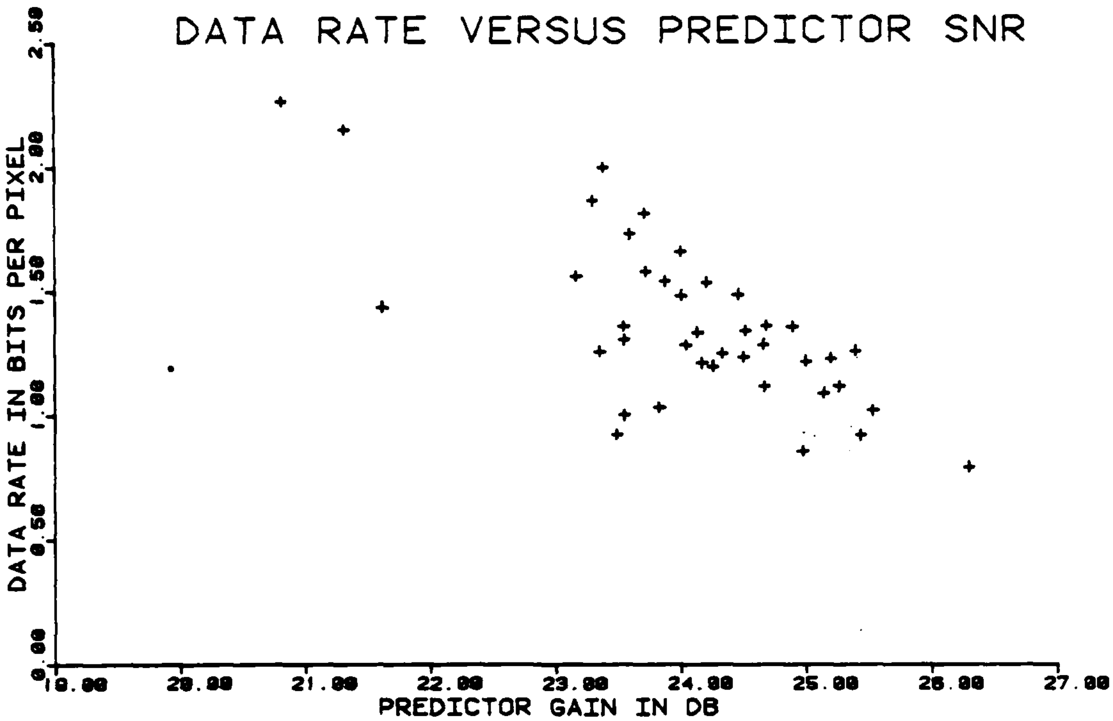


FIGURE 5.13(c) SLOW PHONE 3. Data Rate to Predictor Gain

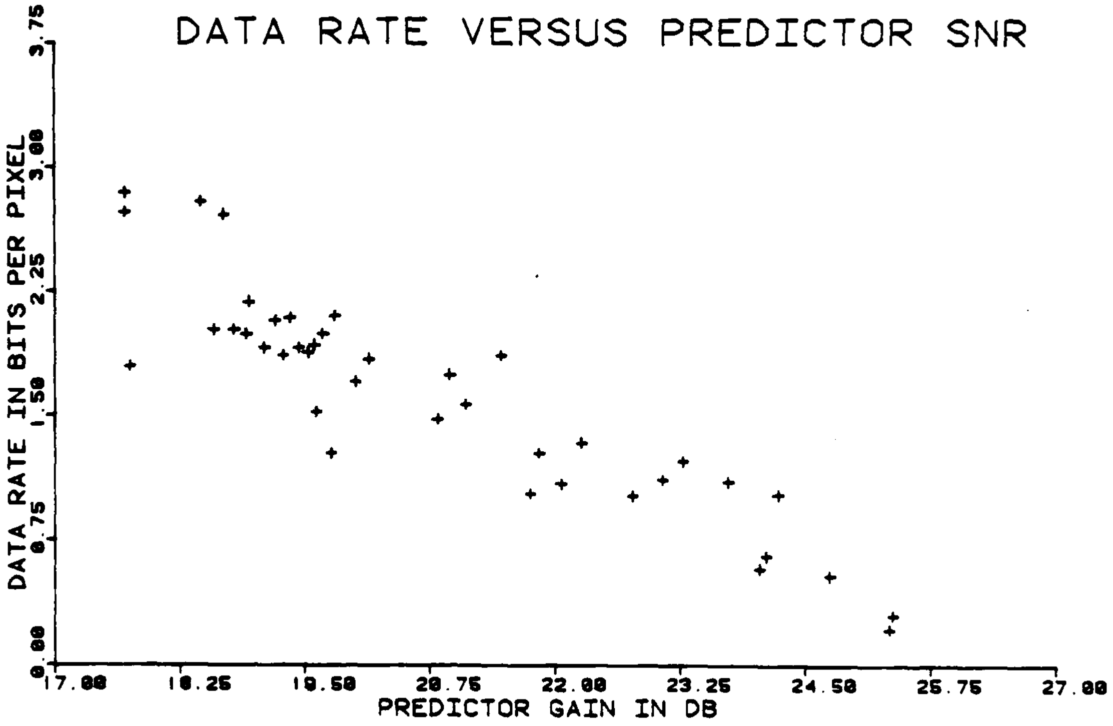


FIGURE 5.13(d) FAST PHONE. Data Rate to Predictor Gain

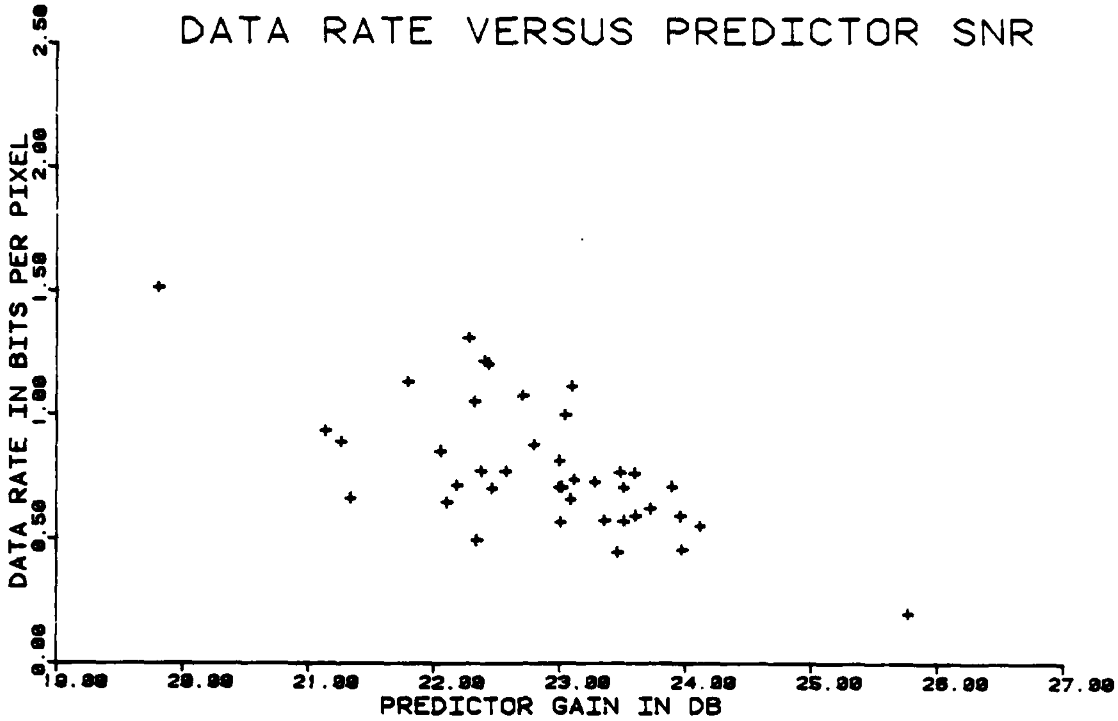


FIGURE 5.13(a) SLOW PHONE 1. Data Rate to Predictor Gain

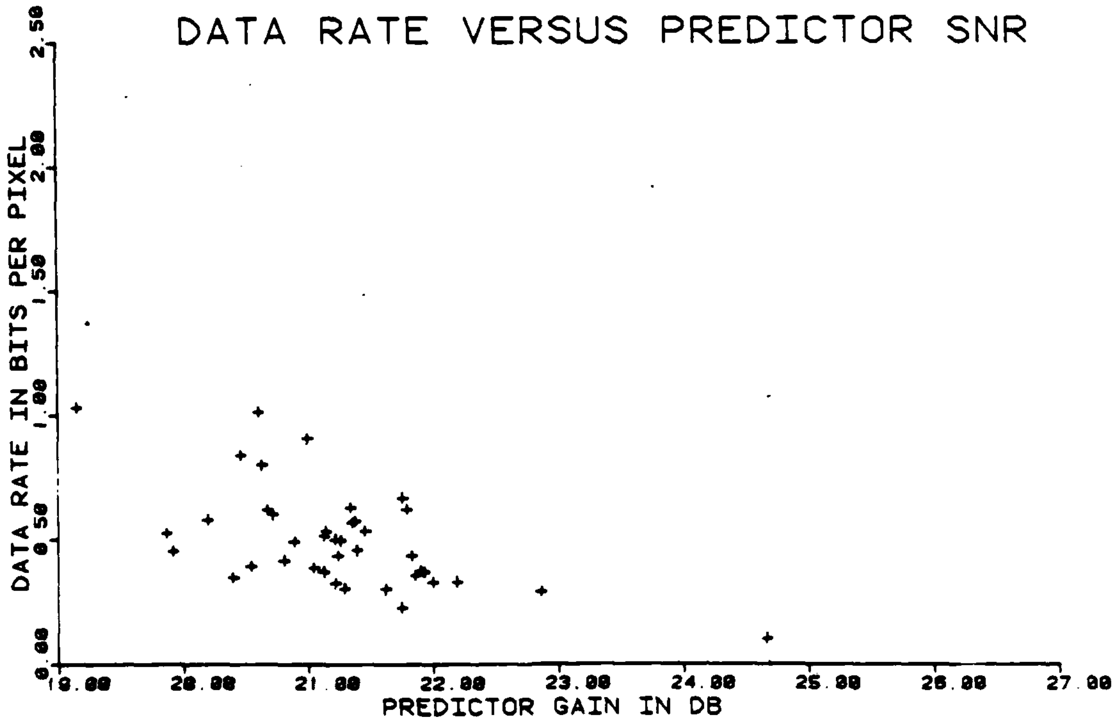


FIGURE 5.13(b) SLOW PHONE 2. Data Rate to Predictor Gain

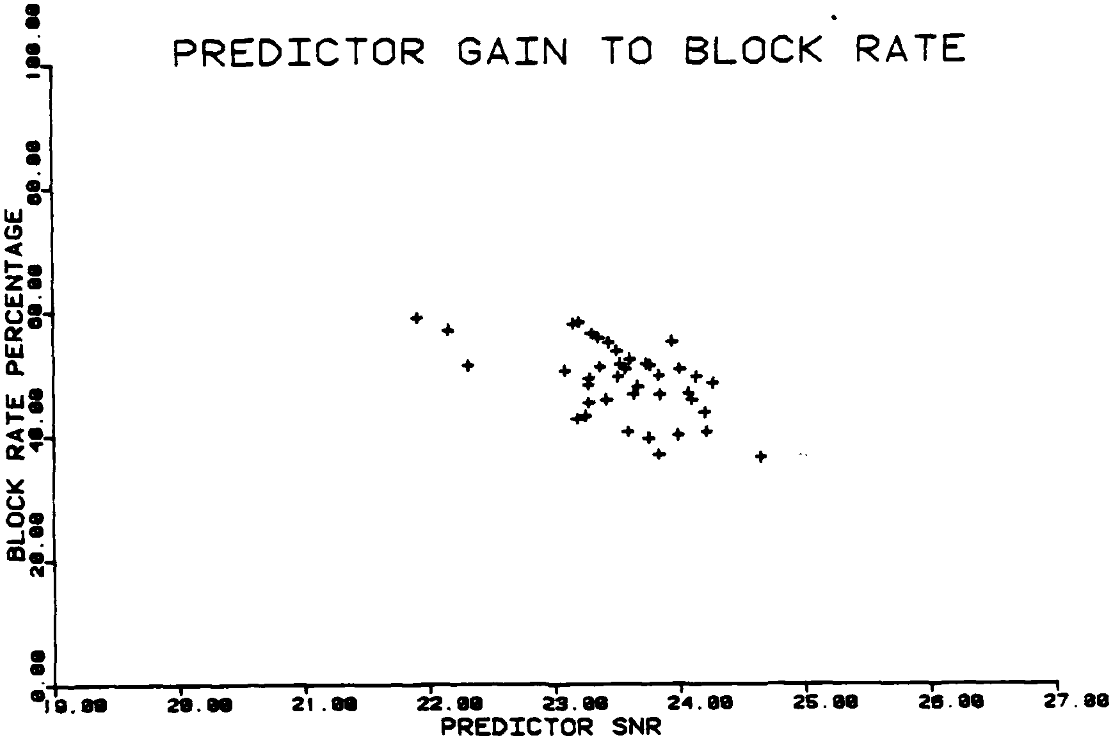


FIGURE 5.12(c) SLOW PHONE 3. Block Rate to Predictor Gain

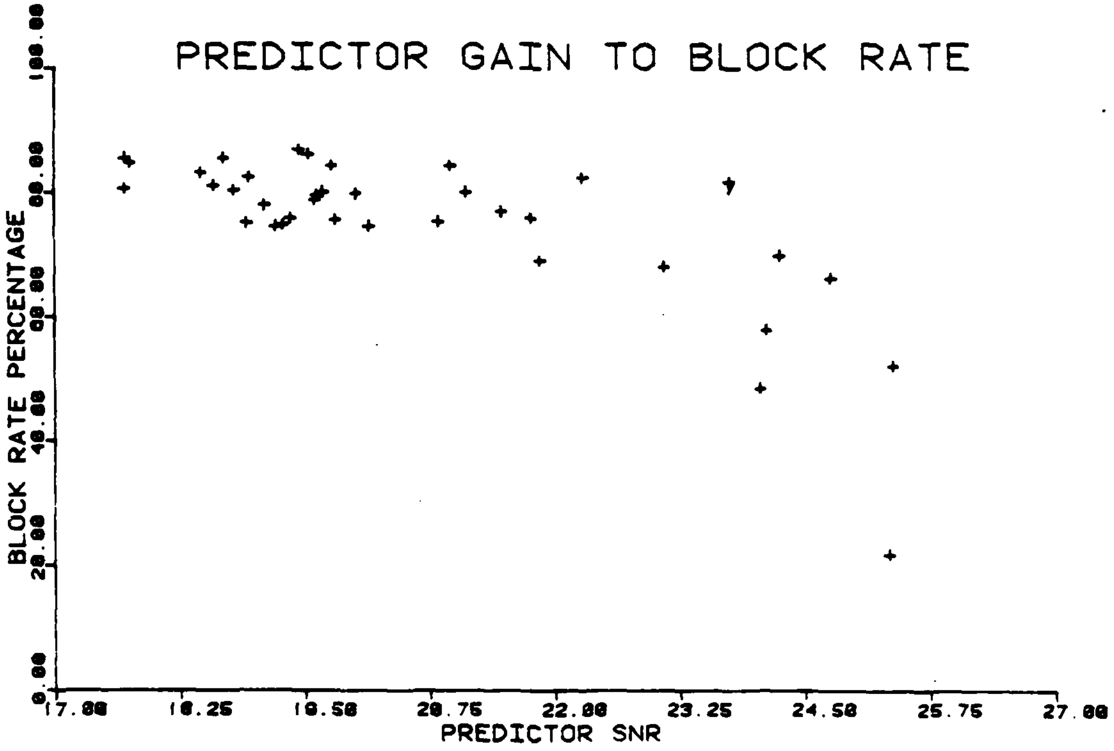


FIGURE 5.12(d) FAST PHONE. Block Rate to Predictor Gain

[55] Jones, R.A., 'Adaptive Hybrid Picture Coding,' Technical Proposal D180-19071-1, Boeing Aerospace Company, Research β Engineering Division, September 1976.

[56] Jones, R.A. and Bowling, C.D., 'An Adaptive Gradient Approach to Displacement Estimation,' NATO Advanced Studies Institute on Image Sequence and Dynamic Scene Analysis,' Braunlage, West Germany, June 1982.

[57] Jones, R.A., Bowling, C.D. and Tejwani, Yogendra, 'Adaptive Hybrid Picture Coding,' Final Scientific Report, Department of Electrical Engineering, University of Arkansas, February 1983.

[58] Kailath, T., 'An Innovations Approach to Least-Squares Estimation. Part I: Linear Filtering in Additive White Noise,' IEEE Transactions on Automatic Control, Volume AC-13, Number 6, December 1968, pp. 646-655.

[59] Kailath, T. and Frost, P., 'An Innovations Approach to Least-Squares Estimation. Part II: Linear Smoothing in Additive White Noise,' IEEE Transactions on Automatic Control, Volume AC-13, Number 6, December 1968, pp. 655-660.

[60] Kanade, Takeo 'Geometrical Aspect of Interpreting Images as a Three-Dimensional Scene,' Proceedings of the IEEE, Volume 71, Number 7, July 1983, pp. 789-802

[61] Kelley, Robert B., Martins, Henrique A.S., Birk, John R., and Dessimoz, Jean-Daniel 'Three Vision Algorithms for Acquiring Workpieces from Bins,' Proceedings of the IEEE, Volume 71, Number 7, July 1983, pp. 803-820.

[62] Kitajima, Hideo, 'Energy Packing Efficiency of the Hadamard Transform,' IEEE Transactions on Communications, Volume 24, Number 11, November 1976, pp. 1256-1258.

[63] Kogo, Toshio, Iijima, Y., and Iinuma, K., 'Statistical Performance Analysis of an Interframe Encoder for Broadcast Television signals,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1868-1876.

[64] Kuo, B.C., Digital Control Systems, Holt, Rinehart and Winston Inc., New York, 1980.

[65] Land, Edwin H., 'The Retinex Theory of Color Vision,' Scientific American, Volume 237, Number 6, December 1977, pp. 108-128.

[66] Legters, George R. Jr., and Young, Tzay Y., 'A Mathematical Model for Computer Image Tracking,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-4, Number 6, November 1982, pp. 583-594.

[67] Levine, Martin D., Youssef, Youssirv M., Noble, Peter, and Boyarsky, Abraham, 'The Quantification of Blood Cell Motion by a Method of Automatic Digital Picture Processing,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 5, September 1980, pp. 444-450.

[68] Liew, Chong Kiew, 'Inequality Constrained Least-Squares Estimation,' Journal of the American Statistical Association, Volume 71, Number 355, September 1976, Theory and Methods Section, pp. 746-751.

[69] Liew, Chong K., and Shim, Jae K., 'A Computer Program for Inequality Constrained Least-Squares Estimation,' Econometrica, Volume 46, Number 1, January 1978, pp. 237.

[70] Limb, J.O., and Murphy, J.A., 'Measuring the Speed of Moving Objects from Television Signals,' IEEE Transactions on Communications, Volume COM-23, Number 4, April 1985, pp. 474-478.

[71] Maxemchuk, N.F., and Stuller J.A., 'An Intraframe DPCM Codec Based Upon Non Stationary Image Model,' Bell System Technical Journal, Volume 58, Number 6 Part II, July-August 1979, pp. 1395-1412.

[72] Maxemchuk, N.F., and Stuller, J.A., 'Reduction of Transmission Error Propagation in Adaptively Predicted, DPCM Encoded Pictures,' Bell System Technical Journal, Volume 58, Number 6 Part II, July-August 1979, pp. 1413-1423.

[73] McCorduck, Pamela, Machines Who Think, W.H. Freeman and Company, San Francisco, 1979.

[74] Meiri, A. Zvi, and Yedilevich, E., 'A Pinned Sine Transform Image Coder,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1728-1735.

[75] Melsa, James L., and Sage, Andrew P., Estimation Theory with Applications to Communications and Control, McGraw-Hill Book Company, 1971.

[76] Melsa, J.L., and Cohn, D.L., Decision and Estimation Theory, McGraw-Hill Book Company, 1978.

[77] Meyers, W., 'Industry Begins to Use Visual Pattern Recognition,' IEEE Computer, Volume 13, Number 5, May 1980, pp. 21-31.

[78] Mitchell, Owen R., and Tavatavai, A.J., 'Channel Error Recovery for Transform Image Coding,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1754-1762.

[79] Mix, Dwight, F., Random Signal Analysis, Addison-Wesley Publishing Company, 1969.

[80] Modestino, James W., and Bhaskaran, V., 'Robust Two-Dimensional Tree Encoding of Images,' IEEE Transactions on Communications, Volume COM-29, Number 12, December 1981, pp. 1786-1798.

[81] Modestino, James W., Daut, D.G., and Vickers, A.L., 'Combined Source-Channel Coding of Images Using Block Cosine Transform,' IEEE Transactions on Communications, Volume COM-29, Number 9, September 1981, pp. 1261-1274.

[82] Modestino, James W., Bhaskaran, V., and Anderson, J.B., 'Tree Encoding of Images in the Presence of Channel Errors,', IEEE Transactions on Information Theory, Volume IT-27, Number 6, November 1981, pp. 677-697.

[83] Mohanty, N.C., 'Computer Tracking of Moving Point Targets in Space,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume Pami-3, Number 5, September 1981, pp. 606-611.

[84] Mood, Alexander M., Graybill, Franklin A., and Boes, Duane C., Introduction to the Theory of Statistics, McGraw-Hill, New York, 1974.

[85] Morrison, Donald F., Multivariate Statistical Methods, McGraw Hill, New York, 1976

[86] Nagel, Hans-Hellmut, 'Representation of Moving Rigid Objects Based on Visual Observations,' IEEE Computer, Volume 14, Number 8, August 1981, pp. 29-39.

[87] Nagy, George, 'Optical Scanning Digitizers,' IEEE Computer, Volume 16, Number 5, May 1983, pp. 13-24.

[88] Natarajan, T. Raj, Ahmed, Nasir, 'On Interframe Transform Coding,' IEEE Transactions on Communications, Volume COM-25, Number 11, November 1977, pp. 1323-1329.

[89] Neter, John and Wasserman, William, Applied Linear Statistical Models, Richard D. Irwin, Inc., Homewood Ill., 1974.

[90] Netravali, A.N., and Robbins, J.D., 'Motion-Compensated Television Coding: Part I,' The Bell System Technical Journal, Volume 58, Number 3, March 1979, pp. 631-669.

[91] Netravali, A.N., and Stuller, J.A., 'Motion-Compensated Transform Coding,' Bell System Technical Journal, Volume 58, Number 7, September 1979, pp. 1703-1718.

[92] Netravali, Arun N., and Limb, John O., 'Picture Coding: A Review,' Proceedings of the IEEE, Volume 68, Number 3, March 1980, pp. 366-406.

[93] Netravali, A.N., and Bowen, E.G., 'Improved Reconstruction of DPCM-Coded Pictures,' Bell System Technical Journal, Volume 61, Number 6, July-August 1982, pp. 969-979.

[94] Neumann, Bernd, 'Exploiting Image Formation Knowledge for Motion Analysis,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 550-554.

[95] Ngan, K.N., 'Adaptive Transform Coding of Video Signals,' IEE Proceedings, Volume 129, Part F, Number 1, February 1982, pp. 28-40.

[96] Ninomiya, Yuichi and Ohtsuka, Yoshimichi, 'A Motion-Compensated Interframe Coding Scheme for Television Pictures,' IEEE Transactions on Communications, Volume COM-30, Number 1, January 1982, Pages 201-211.

[97] O'Leary, Dianne P. and Peleg, Shmuel, 'Digital Image Compression by Outer Product Expansion,' IEEE Transactions on Communications, Volume COM-31, Number 3, March 1983, pp. 441-444.

[98] Oppenheim, Alan V., Schafer, Ronald W., and Stockham, Thomas G., 'Nonlinear Filtering of Multiplied and Convolved signals,' Proceedings of the IEEE, Volume 56, Number 8, August 1968, pp. 1264-1291.

[99] O'Rourke, Joseph and Badler, Norman I., 'Model-Based Image Analysis of Human Motion Using Constraint Propagation,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 522-536.

[100] O'Rourke, J., 'Motion Detection Using Hough Techniques,' IEEE Computer Society Conference on Pattern Recognition and Image Processing, 1981, pp. 82-87.

[101] Ostrem, J.S., and Falconer, D.G., 'A Differential Operator Technique for Restoring Degraded Signals and Images,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-3, Number 3, May 1981, pp. 278-284.

[102] Pack, C.D., and Whitaker, B.A., 'Kalman Filter Models for Network Forecasting,' Bell System Technical Journal, Volume 61, Number 1, January 1982, pp. 1-15.

[103] Papoulis, A., Probability, Random Variables, and Stochastic Processes, McGraw-Hill Book Company, 1965.

[104] Patrick, Edward A., Fundamentals of Pattern Recognition, Prentice Hall, Englewood Cliffs, N.J., 1972.

[105] Peacock, K.L., and Treitel, Sven, 'Predictive Deconvolution: Theory and Practice,' Geophysics, Volume 34, Number 2, April 1969, pp. 155-169.

[106] Pirsh, P., 'Adaptive Intra-Interframe DPCM Coder,' Bell System Technical Journal, Volume 61, Number 5, May-June 1982, pp. 747-764.

[107] Porter, G.B. and Mundy, J.L., 'Visual Inspection System Design,' IEEE Computer, Volume 13, Number 5, May 1980, pp. 40-48.

[108] Prabhu, K.A. and Netravali, A.N., 'Motion Compensated Component Color Coding,' IEEE Transactions on Communications, Volume COM-30, Number 12, December 1982, pp. 2519-2527.

[109] Prabhu, K.A. and Netravali, A.N., 'Motion Compensated Composite Color Coding,' IEEE Transactions on Communications, Volume COM-31, Number 2, February 1983, pp. 216-223.

[110] Pratt, W.K., Digital Image Processing, Wiley Interscience, 1978.

[111] Price, C., Snyder, W., and Rajala, S., 'Computer Tracking of Moving Objects Using a Fourier-Domain Filter Based on a Model of the Human Visual System,' IEEE Computer Society Conference on Pattern Recognition and Image Processing, 1981, pp. 98-102.

[112] Rashid, H.U., 'An Innovations Approach to Displacement Estimation and Image Analysis in Time Varying Images,' PhD Dissertation, Department of Electrical Engineering, University of Arkansas, August 1980.

[113] Rashid, H.U., and Jones, R.A., 'An Adaptive Estimation Approach to Estimation Displacement in Time Varying Images,' SPIE 23rd Internation Symposium, August 1979, San Diego, California.

[114] Regan, David, Beverley, Kenneth and Cynader, Max, 'The Visual Perception of Motion in Depth,' Scientific American, Volume 241, Number 1, July 1979, pp. 136-151.

[115] Roach, John W. and Aggarwal, J.K., 'Determining the Movement of Objects from a Sequence of Images,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 554-562.

[116] Robbins, John D., and Netravali, Arun N., 'Interframe Television Coding Using Movement Compensation,' Internation Conference on Communications Record, ICC'79, Volume 2, pp.23.4.1-23.4.5.

[117] Robbins, J.D., and Netravali, A.N., 'Spatial Subsampling in Motion-Compensated Television Coders,' Bell System Technical Journal, Volume 61, Number 8, October 1982, pp. 1895-1910.

[118] Rock, Irvin, 'Anorthoscopic Perception,' Scientific American, Volume 244, Number 3, March 1981, pp. 145-153.

[119] Roese, John A., Pratt, William K., and Robinson, Guner S., 'Interframe Cosine Transform Image Coding,' IEEE Transactions on Communications, Volume COM-25, Number 11, November 1977, pp. 1329-1339.

[120] Rosenfeld, A., Picture Processing by Computer, Academic Press Inc., 1969.

[121] Rosenfeld, Azriel, and Kak, Avinash, Digital Picture Processing, Academic Press Inc., 1976.

[122] Sabri, Shaker 'Movement-Compensated Interframe Prediction for NTSC Colour TV Signals,' Paper Presented at the Nato Advanced Stuay Institute on Image Sequence Processing and Dynamic Scene Analysis, Braunlage, West Germany, June 1982.

[123] Savol, A.M., Li, C.C., and Hoy, R.J., 'Computer-Aided Recognition of Small Rounded Pnuemoconiosis Opacities in Chest X-Rays,' IEEE PAMI-2, Number 5, pp. 479-482.

[124] Schalkoff, R.J., and McVey, E.S., 'A Model and Tracking Algorithm for a Class of Video Targets,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-4, Number 1, January 1982, pp. 2-10.

[125] Searl, S.R., Linear Models, John Wiley and Sons, New Yc k, 1971.

[126] Shapiro, Linda G., and Haralick, Robert M., 'Organization of Relational Models for Scene Analysis,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-4, Number 6, November 1982, pp. 595-602.

[127] Sternberg, Stanley R., 'Biomedical Image Processing,' IEEE Computer, Volume 16, Number 1, January 1983, pp. 22-34.

[128] Stockham, Thomas G., 'Image Processing in the Context of a Visual Model,' Proceedings of the IEEE, Volume 60, Number 7, July 1972, pp. 828-842.

[129] Stuller, J.A. and Kurz, B., 'Two-Dimensional Markov Representations of Sampled Images,' IEEE Transactions on Communications, Volume COM-24, Number 10, October 1976, pp. 1148-1152.

[130] Stuller, J.A. and Netravali, A.N., 'Transform Domain Motion Estimation,' The Bell System Technical Journal, Volume 58, Number 7, September 1979, pp. 1673-1702.

[131] Stuller, J.A., Netravali, A.N., and Robbins, J.D., 'Interframe Television Coding Using Gain and Displacement Compensation,' Bell System Technical Journal, Volume 59, Number 7, September 1980, pp. 1227-1240.

[132] Tam, Tai On, Stuller, John A., 'Line-Adaptive Hybrid Coding of Images,' IEEE Transactions on Communications, Volume COM-31, Number 3, March 1983, pp. 445-450.

[133] Thompson, William B. and Barnard, Stephen T., 'Lower-Level Estimation and Interpretation of Visual Motion,' IEEE Computer, Volume 14, Number 8, August 1981, pp. 20-28.

[134] Tou, J.T., and Gonzalez, R.C., Pattern Recognition Principles, Addison-Wesley Publishing Company, Reading, Mass., 1974.

[135] Tsotsos, John K., Mylopoulos, John, Covvey, H. Dominic, and Zucker, Steven W., 'A Framework for Visual Motion Understanding,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 563-573.

[136] Tsuji, Saburo, Osada, M., and Yachida, M., 'Tracking and Segmentation of Moving Objects in Dynamic Line Images,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 516-522.

[137] Ullman, Shimon, 'Analysis of Visual Motion by Biological and Computer Systems,' IEEE Computer, Volume 14, Number 8, August 1981, pp. 57-69.

[138] Webb, J.A., and Aggarwal, J.K., 'Visual Interpretation of the Motion of Objects in Space,' IEEE Computer Society Conference on Pattern Recognition and Image Processing, 1981, pp. 516-521.

[139] Webb, John A., Aggarwal, J.K., 'Visually Interpreting the Motion of Objects in Space,' IEEE Computer, Volume 14, Number 8, August 1981, pp. 40-46.

[140] Williams, R.A., and Chang, W.S.C., 'Resolution and Noise in Fourier-Transform Spectroscopy,' Journal of the Optical Society of America, Volume 56, Number 2, February 1966, pp. 167-170.

[141] Williams, Thomas D., 'Depth from Camera Motion in a Real World Scene,' IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, Number 6, November 1980, pp. 511-516.

[142] Wilson, R., Knutsson, H.E., and Granlund, G.H., 'Anisotropic Nonstationary Image Estimation and Its Applications: Part II-Predictive Image Coding,' IEEE Transactions on Communications, Volume COM-31, Number 3, March 1983, pp. 398-406.

[143] Winston, Patrick Henry, The Psychology of Computer Vision, McGraw-Hill Book Company, 1975.

[144] Wintz, Paul A., 'Transform Picture Coding,' Proceedings of the IEEE, Volume 60, Number 7, July 1972, pp. 809-820.

[145] Yachida, Masahiko and Tsuji, Saburo, 'A Versatile Machine Vision System for Complex Industrial Parts,' IEEE Transactions on Computers, Volume C-26, Number 9, September 1977, pp. 882-894.

[146] Yachida, Masahiko and Tsuji, Saburo, 'Industrial Computer Vision in Japan,' IEEE Computer, Volume 13, Number 5, May 1980, pp. 50-62.

[147] Yellot, John I. Jr., 'Binocular Depth Inversion,' Scientific American, Volume 245, Number 1, July 1981, pp. 148-159.

[148] Zadeh, L.A., and Ragazzini, 'An Extension of Wiener's Theory of Prediction,' Journal of Applied Physics, Volume 21, pp. 545-655, July 1950.

[149] Zetterberg, Lars H., Ericsson, Staffan, and Brusewitz, Harald, 'Interframe DPCM with Adaptive Quantization and Entropy Coding,' IEEE Transactions on Communications, Volume COM-30, Number 8, August 1982, pp. 1888-1899.

[150] Zschunke, Willmut, 'DPCM Picture Coding with Adaptive Prediction,' IEEE Transactions on Communications, Volume COM-25, Number 11, November 1977, pp. 1295-1302.

Appendix A

PEL RECURSIVE, COEFFICIENT RECURSIVE AND RESIDUAL RECURSIVE DISPLACEMENT ESTIMATION

PROGRAM DISPEST1

```
C***********************************************************
C*                                                         *
C* PROGRAM DISPEST1   AUTHOR - CARL BOWLING   DATE 9/12/83 *
C*                                                         *
C* PROGRAM FOR DISPLACEMENT ESTIMATION                     *
C*     METHOD IS COEFFICIENT RECURSIVE BY STULLER AND      *
C*     NETRAVALI OF BELL LABORATORIES.  IT MAY ALSO BE USED*
C*     FOR PEL RECURSIVE BY NETRAVALI AND ROBBINS          *
C*                                                         *
C*     SUBROUTINES OR FUNCTIONS REQUIRED -                 *
C*        XFORM  - ROUTINE TO PERFORM TRANSFORMATION       *
C*        PLOTS  - CALCOMP TRANSLATOR AND PLOT SOFTWARE    *
C*        RINTRP - FUNCTION FOR INTEGER INTERPOLATION      *
C*                                                         *
C***********************************************************
C
      REAL V(2,2),H(8,8),CLL(6,8),CP(6,256),CPP(6,8),CPPP(6,8)
      INTEGER IL(6,256),IP(6,256),ILL(6,8),IPP(6,8)
      REAL G(6,8),GG(6,8),CLLL(6,8),ERROR(6,8),DISP(250),XIT(250)
      DATA PI/3.14159265/
C
C     INITIALIZE MATRICES TO VALUE OF 128
C
      DATA IL/1536*128/,IP/1536*128/
C
C     SET UP THE TRANSFORMATION MATRIX IN THE H ARRAY
C
      DATA H/12*1.00,4*-1.00,2*1.00,4*-1.00,4*1.00,2*-1.00,
     1         2*1.00,2*-1.00,  1.00,2*-1.00,2*1.00,2*-1.00,
     2         2*1.00,2*-1.00,  1.00,   -1.00,2*1.00,   -1.00,
     3           1.00,   -1.00,  1.00,2*-1.00,  1.00,   -1.00,
     4         2*1.00,   -1.00,  1.00,   -1.00,  1.00,   -1.00,
     5           1.00,   -1.00/
C
C     METHOD IS USED TO DETERMINE IF PEL RECURSIVE OR COEFFICIENT
C     RECURSIVE DISPLACEMENT ESTIMATION IS USED.  APPROPRIATE LINES
C     MUST ALSO BE COMMENTED OUT DUE TO THE COMPILER USED.
C     METHOD = 0 --> PEL RECURSIVE
C     METHOD = 1 --> COEFFICIENT RECURSIVE
C
C     LOOP IS USED TO DETERMINE IF EVERY LOOP ITERATION OR
C     EVERY BLOCK ITERATION IS TO BE USED.
C     LOOP = 0 --> EVERY BLOCK ITERATION
C     LOOP = 1 --> EVERY LOOP ITERATION
C
C     NPLOT IS A FLAG TO DIRECT THE PLOTTING OF THE DATA
C     NPLOT = 0 --> GO TO 4025 SCREEN
C     NPLOT = 1 --> GO TO 4662 PLOTTER
C
      LOOP = 1
      METHOD = 1
      NPLOT = 0
      SQRT2 = 1./((2.)**.5)
      IF(METHOD.EQ.0)SQRT2 = 0.0
      V(1,1) =  SQRT2
      V(1,2) =  V(1,1)
      V(2,1) =  V(1,1)
      V(2,2) = -V(1,1)
      FAC = .5*SQRT2
      DO 10 I=1,8
      DO 10 J=1,8
      H(I,J) = H(I,J)*FAC
 10   CONTINUE
      IF(METHOD.NE.0) GO TO 40
      DO 20 I=1,8
 20   H(I,I) = 1.0
      DO 30 I=1,2
 30   V(I,I) = 1.0
 40   CONTINUE
```

```
C***************************************************************
C*      GENERATE THE TEST ARRAYS IN IP AND IL                  *
C*      IP IS THE PRESENT IMAGE FRAME                          *
C*      IL IS THE PREVIOUS IMAGE FRAME                         *
C*      THE TEST ARRAY IS A RADIALLY DECAYING FREQUENCY        *
C*      MODULATED COSINE FUNCTION.                             *
C***************************************************************
C
        DO 50 I=1,6
        DO 50 J=1,60
        R=SQRT( FLOAT( (I+7)**2 + J*J ) )
        IF ( R .GT. 60.0 ) GO TO 50
        P=(1. - R/60.)*10. + 10.
        IPT=100. * EXP(-.01*R) * COS(2.*PI*R/P) + 128.
        IP(I,J+128)=IPT
        IP(I,129-J)=IPT
        IL(I,J+126)=IPT
        IL(I,127-J)=IPT
   50   CONTINUE
C
C***************************************************************
C*                                                             *
C*     TRANSFORM THE ORIGINAL ARRAY WITH SUBROUTINE XFORM      *
C*                                                             *
C***************************************************************
C       NR - THE NUMBER OF ROWS IN TRANSFORMATION
C       NC - THE NUMBER OR COLUMNS IN TRANSFORMATION
        NR=6
        NC=8
C
C       TRANSFORM IS (V)*(I)*(H)
C
        DO 80 J=1,256,8
C
C       J IS THE BLOCK NUMBER
C
        DO 60 K1 = 1,6
C
C       K1 IS THE ROW NUMBER
        J7 = J + 7
        K3 = 0
        DO 60 K2 = J,J7
        K3 = K3 + 1
        CPPP(K1,K3) = IP(K1,K2)
   60   CONTINUE
        CALL XFORM(CPP,V,CPPP,H,NR,NC)
        DO 70 K1 = 1,6
        K3 = 0
        DO 70 K2=J,J7
        K3 = K3 + 1
        CP(K1,K2) = CPP(K1,K3)
   70   CONTINUE
   80   CONTINUE
C
C***************************************************************
C*                                                             *
C*      END TRANSFORMATION - START OF DISPLACEMENT ITERATION*
C*                                                             *
C***************************************************************
C
C       EPSIL - GAIN FACTOR (READ IN INTERACTIVELY)
C       DA IS THE ACTUAL FRAME DISPLACEMENT IN PIXELS
C       DI IS THE ESTIMATE OF THE PIXEL DISPLACEMENT
C       ITNUM IS THE ITERATION LOOP COUNTER
C
        DA = 2.0
        XOFF = 1.35
        YOFF = 1.5
        IF(NPLOT.EQ.0)CALL PLOTS(IBUF,1,15)
        IF(NPLOT.EQ.0)XOFF = .1
        IF(NPLOT.EQ.0)YOFF = .1
   90   CONTINUE
        WRITE(6,500)
        READ(5,510)EPSIL
        IF(EPSIL.GT.1) STOP
```

```
      IF(NPLOT.EQ.0)CALL ERASE
      DI = 0.0
C
C     I IS THE BLOCK NUMBER (USE ONLY 10TH - 23RD)
C
      ITNUM = 0
      DO 150 I=10,23
C
C     JJ - START COLUMN LOCATION OF CURRENT BLOCK
C     JD - START COLUMN LOCATION OF PREVIOUS BLOCK
C
      JJ = (I-1)*8 + 1
      J = 1
      K = 3
      IF(LOOP.EQ.0) GO TO 95
C
C     IF LOOP SET TO 0, COMMENT OUT THE FOLLOWING TWO LINES
C     SOME COMPILERS MAY NOT ALLOW THE JUMP AROUND DO LOOPS
C
      DO 140 J=1,8
      DO 140 K=3,4
   95 RD = FLOAT(JJ) - DI
      JD = RD
      RDIFF = RD - FLOAT(JD)
C
C****************************************************
C*                                                  *
C*    CALCULATE INTERPOLATED DISPLACED BLOCK        *
C*                                                  *
C****************************************************
C
      JD7 = JD+7
      JDD = 0
      DO 100 L=JD,JD7
      JDD = JDD + 1
      DO 100 M=1,6
      X = RI(IL(M,L),IL(M,L+1),RDIFF)
  100 CLLL(M,JDD) = X
C
C****************************************************
C*                                                  *
C*    END OF FRAME INTERPOLATION DETERMINATION -    *
C*    FIND THE FRAME ERROR                          *
C*                                                  *
C****************************************************
C
      CALL XFORM(CLL,V,CLLL,H,NR,NC)
      DO 110 L=1,8
      DO 110 M=3,4
      ERROR(M,L) = CP(M,L+JJ-1) - CLL(M,L)
  110 CONTINUE
C
C****************************************************
C*                                                  *
C*    FIND TIME DOMAIN GRADIENT OF THE DISPLACED FRAME  *
C*    CLLL.  GRADIENT FOUND BY CENTRAL DIFFERENCE METHOD *
C*                                                  *
C****************************************************
C
      DO 130 L=3,4
      DO 120 M=2,7
      GRAD1 = CLLL(L+1,M) - CLLL(L-1,M)
      GRAD1 = 0.
      GRAD2 = CLLL(L,M+1) - CLLL(L,M-1)
  120 G(L,M) = (GRAD1 + GRAD2) * .5
      GRAD1 = (CLLL(L,2) - CLLL(L,1))*2.
      GRAD2 = (CLLL(L+1,1) - CLLL(L-1,1))
      GRAD2 = 0.
      G(L,1) = (GRAD1 + GRAD2) *.5
      GRAD1 = (CLLL(L,8) - CLLL(L,7))*2.
      GRAD2 = (CLLL(L+1,8) - CLLL(L-1,8))
      GRAD2 = 0.
      G(L,8) = (GRAD1 + GRAD2)*.5
  130 CONTINUE
      CALL XFORM(GG,V,G,H,6,8)
```

```
C
C***********************************************************
C*                                                         *
C*     GENERATE THE DISPLACEMENT ESTIMATE                  *
C*                                                         *
C***********************************************************
C
      IF(LOOP.EQ.1) GO TO 135
C
C     IF LOOP IS SET TO 1, COMMENT OUT THE FOLLOWING TWO LINES
C     SOME COMPILERS MAY NOT ALLOW THE JUMP AROUND DO LOOPS
C
C     DO 140 J = 1,8
C     DO 140 K = 3,4
  135 DI = DI - EPSIL*ERROR(K,J)*GG(K,J)
      ITNUM = ITNUM + 1
      XIT(ITNUM) = ITNUM
      DISP(ITNUM) = DA - DI
  140 CONTINUE
  150 CONTINUE
      IF(NPLOT.EQ.1)CALL PLOTS(IBUF,1,15)
      CALL PLOT(XOFF,YOFF,-3)
      IF(NPLOT.EQ.0)XOFF = 0.0
      IF(NPLOT.EQ.0)YOFF = 0.0
      CALL FACTOR(.65)
      IF(NPLOT.EQ.0)CALL FACTOR(.25)
      CALL SCALE(DISP,5.,80,1)
      CALL SCALE(XIT,8.,80,1)
      CALL AXIS(0.0,0.0,'ITERATION NUMBER',-16,8.,0.0,XIT(81),XIT(82))
      CALL AXIS(0.,0.,'DISPLACEMENT ERROR',18,5.,90.,DISP(81),DISP(82))
      IF(NPLOT.EQ.0) GO TO 155
C     IF(NPLOT.EQ.1)CALL NEWPEN('BLAC')
      WRITE(6,520)
  520 FORMAT(1X,'/* HIT ANY SINGLE DIGIT NUMBER AND RETURN IF READY')
      READ(5,530)IANS
  530 FORMAT(I2)
C     IF(NPLOT.EQ.1)CALL NEWPEN('BLAC')
  155 CONTINUE
      CALL LINE(XIT,DISP,80,1,1,3)
      CALL SYMBOL(6.3,4.45,.20,44,0.,-1)
      CALL SYMBOL(6.5,4.5,.14,'=',0.,1)
      CALL NUMBER(6.7,4.5,.14,EPSIL,0.0,5)
      IF(LOOP.EQ.1)CALL SYMBOL(2.,5.,.2,'EVERY LOOP ITERATION ',0.,21)
      IF(LOOP.EQ.0)CALL SYMBOL(2.,5.,.2,'EVERY BLOCK ITERATION',0.,21)
      IF(METHOD.EQ.1)CALL SYMBOL(2.,5.3,.2,'COEFFICIENT RECURSIVE',0.
     #,21)
      IF(METHOD.EQ.0)CALL SYMBOL(2.,5.3,.2,'PEL RECURSIVE',0.,13)
      IF(NPLOT.EQ.1) GO TO 90
      CALL TSEND
      CALL ANMODE
      GO TO 90
  160 CALL PLOT(10.,10.,999)
  500 FORMAT(1X,'/*','WHAT VALUE FOR THE GAIN EPSILON?')
  510 FORMAT(F10.6)
      STOP
      END
```

```
                              PROGRAM RESDISP
PRT FILE 5279 FROM VS1      COPY 001   NOHOLD

C********************************************************
C*                                                      *
C*     PROGRAM BY - CARL BOWLING                        *
C*     LATEST UPDATE - OCTOBER 26, 1983                 *
C*                                                      *
C*     PROGRAM FOR DISPLACEMENT ESTIMATION              *
C*     METHOD IS RESIDUAL RECURSIVE                     *
C*                                                      *
C*     SUBROUTINES OR FUNCTIONS REQUIRED -              *
C*                                                      *
C*     XFORMB - PERFORMS A 1 OR 2 DIMENSIONAL UNITARY   *
C*              TRANSFORMATION ON THE INPUT ARRAY.      *
C*     COMP   - ADAPTIVE HYBRID PICTURE CODING DATA     *
C*              COMPRESSION ROUTINE.                    *
C*     PLOTS  - ALL THE CALCOMP DRIVER SOFTWARE AND THE *
C*              TRANSLATOR SOFTWARE FOR THE TEKTRONIX   *
C*              4662 PLOTTER AND 4025 DISPLAY.          *
C*     RINTRP - FUNCTION TO INTERPOLATE BETWEEN ADJACENT *
C*              INTEGER VALUES.                         *
C*                                                      *
C********************************************************
C
      EXTERNAL HADGEN
      REAL DLIF(8,8),DLTF(8,8),ERROR(8,8),CP(8,256)
      REAL COEFF(8,3),CL(8,256),XIT(250),DISP(250)
      INTEGER IL(8,256),IP(8,256)
      COMMON /CODPRM/ IR,IC,IBSZ,NTYPE,OBSZ,OTYPE,IBLKSZ
      COMMON /FT/ KFFT
      DATA PI/3.14159265/
      DATA NTYPE1/'IN*4'/,OTYPE1/'REAL'/
C
C     NTYPE1 IS THE DATA TYPE OF THE INPUT ARRAY (INTEGER*4)
C     OTYPE1 IS THE DATA TYPE OF THE OUTPUT ARRAY (REAL*4)
C
C     INITIALIZE PAST AND PRESENT FRAME MATRICES TO VALUE OF 128
C
      DATA IL/2048*128/,IP/2048*128/
C
C     SET CONSTANTS FOR TRANSFORM
C
      NTYPE = NTYPE1
      OTYPE = OTYPE1
      KFFT  = 0
      IBLKSZ= 8
      IBSZ  = 8
      OBSZ  = 8
C
C********************************************************
C*                                                      *
C*     GENERATE THE TEST ARRAYS IN IP AND IL            *
C*     IP IS THE PRESENT IMAGE FRAME                    *
C*     IL IS THE PREVIOUS IMAGE FRAME                   *
C*     THE TEST ARRAY IS A RADIALLY DECAYING FREQUENCY  *
C*     MODULATED COSINE FUNCTION.  ONLY A PORTION OF THE *
C*     ENTIRE IMAGE IS USED.                            *
C*                                                      *
C********************************************************
C
      DO 10 I=1,8
      DO 10 J=1,60
      R=SQRT( FLOAT( (I+7)**2 + J*J ) )
      IF ( R .GT. 60.0 ) GO TO 10
      P=(1. - R/60.)*10. + 10.
      IPT=100.*EXP(-.01*R)*COS(2.*PI*R/P) + 128.
      IP(I,J+128)=IPT
      IP(I,129-J)=IPT
      IL(I,J+126)=IPT
      IL(I,127-J)=IPT
   10 CONTINUE
```

```
C***********************************************************
C*                                                         *
C*    START OF DISPLACEMENT ITERATION                      *
C*                                                         *
C***********************************************************
C
C     EPSIL - GAIN FACTOR (READ IN INTERACTIVELY)
C     DA IS THE ACTUAL FRAME DISPLACEMENT IN PIXELS
C     DI IS THE ESTIMATE OF THE PIXEL DISPLACEMENT
C     ITNUM IS THE ITERATION LOOP COUNTER
C
      DA = 2.0
   20 WRITE(6,30)
   30 FORMAT(1X,'/*','WHAT VALUE FOR EPSILON?  F10.6 FORMAT')
      READ(5,40)EPSIL
   40 FORMAT(F10.6)
C
C     CHECK IF GAIN IS GREATER THAN 1.  IF SO STOP PROGRAM
C
      IF(EPSIL.GT.1.) GO TO 130
      CALL PLOTS(IBUF,1,9)
      DI = 0.0
C
C***********************************************************
C*                                                         *
C*    I     - THE BLOCK NUMBER (USE ONLY 10TH - 23RD)      *
C*    ITNUM - THE ITERATION NUMBER                         *
C*    JJ    - START COLUMN LOCATION OF CURRENT BLOCK       *
C*    JD    - START COLUMN LOCATION OF PREVIOUS BLOCK      *
C*                                                         *
C***********************************************************
C
      ITNUM = 1
      XIT(1) = 1.
      DISP(1) = 2.0
      DO 120 I=10,23
      JJ = (I-1)*8 + 1
      DO 120 J = 1,8
      DO 120 K = 3,3
      RD = FLOAT(JJ) - DI
      JD = RD
      RDIFF = RD - FLOAT(JD)
C
C***********************************************************
C*                                                         *
C*    CALCULATE INTERPOLATED DISPLACED BLOCK               *
C*    IN THE TIME DOMAIN                                   *
C*                                                         *
C***********************************************************
C
      JD7 = JD+7
      JDD = 0
      DO 50 L=JD,JD7
      JDD = JDD + 1
      DO 50 M=1,8
      X = RINTRP(IL(M,L),IL(M,L+1),RDIFF)
   50 DLIF(M,JDD) = X
      DO 60 L=1,8
   60 WRITE(1)(DLIF(L,M),M=1,8)
      REWIND 1
C
C***********************************************************
C*                                                         *
C*    END OF DISPLACED BLOCK INTERPOLATION                 *
C*                                                         *
C*    CALCULATE THE RESIDUAL SEQUENCE FOR USE AS THE       *
C*    GRADIENT.                                            *
C*                                                         *
C***********************************************************
C
      NTYPE = OTYPE1
      IR = 0
      IC = +1
      CALL XFORMB(HADGEN,8,8,1,2,DLTF)
      REWIND 2
      DO 70 L=1,8
   70 READ(2)(DLTF(L,M),M=1,8)
      REWIND 2
      CALL COMP(DLTF,8,8,3,COEFF)
      REWIND 1
      DO 80 L=1,8
   80 WRITE(1)(DLTF(L,M),M=1,8)
      REWIND 1
      IR = 0
      IC = -1
      CALL XFORMB(HADGEN,8,8,1,2,DLTF)
      REWIND 1
      REWIND 2
      DO 90 L=1,8
   90 READ(2)(DLTF(L,M),M=1,8)
      DO 100 L=1,8
      DO 100 M=1,8
  100 DLTF(L,M) = DLTF(L,M)/FLOAT(IBLKSZ)
      REWIND 1
      REWIND 2
C
C***********************************************************
C*                                                         *
C*    END OF RESIDUAL CALCULATION                          *
C*                                                         *
C*    FIND THE TIME DOMAIN FRAME ERROR                     *
C*                                                         *
C***********************************************************
C
      DO 110 L=1,8
      DO 110 M=3,3
  110 ERROR(M,L) = FLOAT(IP(M,L+JJ-1)) - DLIF(M,L)
C
C***********************************************************
C*                                                         *
C*    GENERATE THE DISPLACEMENT ESTIMATE                   *
C*                                                         *
C***********************************************************
C
      DI = DI - EPSIL*ERROR(K,J)*DLTF(K,J)
      ITNUM = ITNUM + 1
      XIT(ITNUM) = ITNUM
      DISP(ITNUM) = 2.0 - DI
  120 CONTINUE
C
C     ALL OF THE CALLS BELOW ARE USED TO PLOT THE RESULTS
C
      CALL PLOT(1.35,6.0,-3)
      CALL FACTOR(.65)
      CALL SCALE(DISP,5.,80,1)
      CALL SCALE(XIT,8.,80,1)
      CALL AXIS(0.0,0.0,'ITERATION NUMBER',-16,8.,0.0,XIT(81),XIT(82))
      CALL AXIS(0.,0.,'DISPLACEMENT ERROR',18,5.,90.,DISP(81),DISP(82))
      CALL LINE(XIT,DISP,80,1,1,3)
      CALL SYMBOL(6.3,4.45,.20,44,0.,-1)
      CALL SYMBOL(6.5,4.5,.14,'=',0.,1)
      CALL NUMBER(6.7,4.5,.14,EPSIL,0.0,5)
      CALL SYMBOL(1.0,5.0,.28,'DISPLACEMENT ESTIMATION',0.,23)
      CALL SYMBOL(2.0,5.4,.20,'RESIDUAL RECURSIVE',0.,18)
      CALL PLOT(10.,10.,2)
      CALL TSEND
      CALL ANMODE
      GO TO 20
  130 CALL PLOT(10.,10.,999)
      STOP
      END
```

```
C
      IERSUM = 0
      IERSQR = 0
      K = ISTY - 1
      DO 250 I=1,16
      K = K + 1
      L = IX - 1
      DO 250 J=1,16
      L = L + 1
      IER = B(K,L) - ERROR(I,J)
      EST(K,L) = ERROR(I,J)
      ERRSIG(K,L) = ER16(I,J)
      IERSUM = IERSUM + IER
      IERSQR = IERSQR + IER*IER
      IER = ER16(I,J) + 128
      IF(IER.GT.255)IER=255
      IF(IER.LT.1) IER = 1
      HIST(IER) = HIST(IER) + 1.
  250 CONTINUE
C
C     CALCULATE THE MEAN AND VARIANCE OF THE QUANTIZED VERSION FOR
C     COMPARISON.
C
      A1 = IERSUM
      S1 = SQRT((FLOAT(IERSQR) - (A1*A1/256.))/255.)
      A1 = A1/256.
      IF(IPRT.EQ.2)WRITE(6,1060)IBLK,JBLK,MIN,(LOC(KC),KC=1,2),AVG,STD,
     # A1,S1,INTBIT,NBITS
  260 CONTINUE
C
C     CALCULATE THE SIGNAL TO NOISE RATIO
C
      ITOTAL = 0
      DO 270 I=1,ISIZE
      DO 270 J=1,JSIZE
      ITOTAL = ITOTAL + B(I,J)
  270 CONTINUE
      TOTAL = FLOAT(ITOTAL)
      XMEAN = TOTAL/SIZE
      SSE = 0.0
      SSS = 0.0
      SSP = 0.0
      DO 290 J=1,JSIZE
      PSSS = 0.0
      PSSE = 0.0
      PSSP = 0.0
      DO 280 I=1,ISIZE
      PSSP = PSSP + ERRSIG(I,J)*ERRSIG(I,J)
      IERR = B(I,J) - EST(I,J)
      INTER = B(I,J)
      SIG = FLOAT(INTER) - XMEAN
      PSSS = PSSS + SIG*SIG
      PSSE = PSSE + FLOAT(IERR*IERR)
  280 IA(I,J) = EST(I,J)
      SSS = SSS + PSSS
      SSP = SSP + PSSP
  290 SSE = SSE + PSSE
      IF(IPRT.EQ.2)WRITE(6,1070)SSE,SSS
      SNR = 10.*ALOG10(SSS/SSE)
      SNRP = 10.*ALOG10(SSS/SSP)
      SNRARY(IMAGE) = SNR
      SNRPAR(IMAGE) = SNRP
      IF(IPRT.EQ.1)WRITE(6,1080)SNR,SNRP
C
C     WRITE OUTPUT IMAGE OUT TO UNIT 10
C
      CALL COMTAL(EST,ISIZE,JSIZE,10)
C     WRITE(6,1090)FF
      IF(IPRT.GE.1)WRITE(6,1100)
C
C     CALCULATE MOTION RATE (#BLOCKS IN MOTION)
C
      NTAB = 0
      DO 330 I=1,24
      DO 325 K=1,16
```

```
C*      NON-INTEGER DISPLACEMENT ASSUMED AT THIS POINT          *
C*                                                              *
C*      LOCATE THE MINIMUM QUADRANT WITH RESPECT TO THE         *
C*      INTEGER DISPLACEMENT.                                   *
C*                                                              *
C***************************************************************
C
      CALL FILL(IBLK,JBLK,I1M1,J1M1,8,MTAB,24,16,'RR')
      INTBIT = 1
      CALL LOCATE(LOC,F,21,21,MIN)
      NBITS = NBITS + 21 + 4*NPBITS
C
C     SET UP THE REGRESSION PROBLEM
C
      K = 0
      DO 190 L=1,8
      I = L + IXOFF
      DO 190 M=1,8
      J = M + IYOFF
      K = K + 1
      DO 180 N=1,4
      X(K,N) = D8(J+QOFF(MIN,N,1),I + QOFF(MIN,N,2))
  180 CONTINUE
      OUT(K) = C8(M,L)
  190 CONTINUE
C
C---------------------------------------------------------------
C|                                                             |
C|     CALCULATE THE NEW PREDICTION COEFFICIENTS               |
C|                                                             |
C---------------------------------------------------------------
C
      CALL COEFGN(X,OUT,RCOEF,K)
C
C     QUANTIZE THE PREDICTOR COEFFICIENTS
C
      DO 200 I=1,4
  200 RCOEF(I) = (AINT(RCOEF(I)*RLEVEL + .5))/RLEVEL
      IF(K.NE.129) GO TO 210
      RCOEF(1) = 0.0
      RCOEF(2) = 0.0
      RCOEF(3) = 1.0
      RCOEF(4) = 0.0
      MIN = 1
  210 CONTINUE
C
C***************************************************************
C*                                                             *
C*    CALCULATE THE PREDICTED IMAGE BLOCK                      *
C*                                                             *
C***************************************************************
C
      CALL PRED(C8,D8,E8,28,8,IXOFF,IYOFF,VA,AV,QOFF,MIN,RCOEF,INTBIT,
     # NBITS,RETCOD,ER8,DIST,LVLARA)
      IF(IPRT.EQ.2)WRITE(6,1050)I1,J1,MIN,LOC(1),LOC(2),AV,VA,INTBIT,
     #  NBITS,RETCOD
      IF(RETCOD.EQ.1)NBITS = NBITS - 21
      IF(RETCOD.EQ.1) GO TO 170
C
C     WRITE ESTIMATE AND ERROR OUT TO 16X16 ARRAY
C
      K = IBLK1
      DO 220 I=1,8
      K = K + 1
      L = JBLK1
      DO 220 J=1,8
      L = L + 1
      ERROR(K,L) = E8(I,J)
      ER16(K,L) = ER8(I,J)
  220 CONTINUE
  230 CONTINUE
  240 CONTINUE
C
C     WRITE QUANTIZED PREDICTION VALUE OUT TO ESTIMATED MATRIX
C     CALCULATE THE MEAN AND VARIANCE OF THE PREDICTION ERROR
```

```
      C8(K,L) = C(I,J)
  140 CONTINUE
      IAVG = IFIX(FLOAT(ITOTAL)/64. + .5)
      CALL INITI(D8,28,28,IAVG)
  150 CONTINUE
C
C     TRANSFER DATA FROM D -> D8
C
      ISTX = IX - 10 + JBLK1
      ISPX = ISTX + 27
      ITY = ISTY - 10 + IBLK1
      IPY = ITY + 27
      ISX = ISTX
      IHX = ISPX
      ISY = ITY
      IHY = IPY
      IF(IBLOCK.EQ.0) GO TO 155
      IF(ISX.LT.1) ISX = 1
      IF(IHX.GT.ISIZE) IHX = ISIZE
      IF(ISY.LT.1) ISY = 1
      IF(IHY.GT.ISIZE) IHY = ISIZE
  155 M = 0
      N = 0
      IF(ISTX.LT.1) M = 1 - ISTX
      IF(ITY.LT.1) N = 1 - ITY
      KCON8(1) = N + 1
      KCON8(2) = M + 1
      K = M
      DO 160 J=ISX,IHX
      K = K + 1
      L = N
      DO 160 I=ISY,IHY
      L = L + 1
      D8(L,K) = IA(I,J)
  160 CONTINUE
C
C-----------------------------------------------------------
C|                                                         |
C|    CALCULATE THE METRIC FOR THE 8 BY 8 BLOCKS           |
C|                                                         |
C-----------------------------------------------------------
C
C     IF(IBLOCK.EQ.1) CALL INITI(F,21,21,0)
      CALL METRIC(C8,D8,F,28,21,8,IMIN,LOC,IBLOCK,KCON8,ITHRS8,
     # 11,11)
      IXOFF = LOC(2) - 1
      IYOFF = LOC(1) - 1
C
C     TEST FOR INTEGER DISPLACEMENT
C
      IF(IMIN.GT.ITHRS8) GO TO 170
      INTBIT = 0
C
C**********************************************************
C*                                                        *
C*    INTEGER DISPLACEMENT ASSUMED AT THIS POINT          *
C*                                                        *
C**********************************************************
C
      RCOEF(1) = 0.0
      RCOEF(2) = 0.0
      RCOEF(3) = 1.0
      RCOEF(4) = 0.0
      MIN = 1
C
C     TEST FOR NO DISPLACEMENT
C
      IF(IXOFF.EQ.10 .AND. IYOFF.EQ. 10) GO TO 210
      CALL FILL(IBLK,JBLK,I1M1,J1M1,8,MTAB,24,16,'II')
      NBITS = NBITS + 20
      GO TO 210
  170 CONTINUE
C
C**********************************************************
C*                                                        *
```

```
      AVG = AV
C
C--------------------------------------------------------
C|                                                      |
C|     PERFORM TEST TO DETERMINE IF SUB-BLOCK PROCESSING |
C|     MAY HELP.  TEST VARIANCE OF ERROR.                |
C|                                                      |
C--------------------------------------------------------
C
      IF(VA.LT.STDERR .AND. ABS(AV).LT.AVGERR) GO TO 240
C
C********************************************************
C*                                                      *
C*   **** *   * ****      ****  *      ***   **** *   * *
C*  *     *   * *   *     *   * *     *   * *     *  *  *
C*  *     *   * *   *     *   * *     *   * *     * *   *
C*   ***  *   * ****  *** ****  *     *   * *     ***   *
C*      * *   * *   *     *   * *     *   * *     * *   *
C*      * *   * *   *     *   * *     *   * *     *  *  *
C*  ****   ***  ****      ****   ****  ***   **** *   * *
C*                                                      *
C***********************   ******************************
C
C     RESET DISPLACEMENT MAP
C
      CALL FILL(IBLK,JBLK,0,0,16,MTAB,24,16,'  ')
C
C     CORRECT DATA RATE FOR SUB-BLOCK PROCESSING
C
      IF(INTBIT.EQ.1) NBITS = NBITS - 18 - 4*NPBITS
      IF(INTBIT.EQ.0.AND.(IXOFF.NE.10 .OR. IYOFF.NE.10))NBITS=NBITS- 18
      DO 230 I1=1,2
      IBLK1 = (I1-1)*8
      DO 230 J1=1,2
      JBLK1 = (J1-1)*8
      I1M1 = I1 - 1
      J1M1 = J1 - 1
      KCON8(1) = 1
      KCON8(2) = 1
      KCON8(3) = 28
      KCON8(4) = 28
C
C--------------------------------------------------------
C|                                                      |
C|     START DATA TRANSFER FOR 8 BY 8 BLOCKS            |
C|                                                      |
C--------------------------------------------------------
C
      IF(IBLOCK.EQ.1) GO TO 130
C
C     SECTION FOR NON-BORDER BLOCK TRANSFER
C
      I = IBLK1
      DO 120 K=1,8
      I = I + 1
      J = JBLK1
      DO 120 L=1,8
      J = J + 1
      C8(K,L) = C(I,J)
  120 CONTINUE
      GO TO 150
  130 CONTINUE
C
C     SECTION FOR BORDER BLOCK TRANSFER
C
      ITOTAL = 0
      I = IBLK1
      DO 140 K=1,8
      M = ISTY + I
      I = I + 1
      J = JBLK1
      DO 140 L=1,8
      N = IX + J
      J = J + 1
      ITOTAL = ITOTAL + IA(M,N)
      CALL FILL(IBLK,JBLK,0,0,16,MTAB,24,16,'RR')
      CALL LOCATE(LOC,F,21,21,MIN)
C
C     QUADRANT NUMBER NOW CONTAINED IN MIN
C
C------------------------------------------------------------
C|                                                          |
C|    CALCULATE THE NEW SET OF PREDICTOR COEFFICIENTS      |
C|    FOR THE CURRENT BLOCK.                               |
C|                                                          |
C|    THE FOUR VALUES ARE CONTAINED IN ARRAY RCOEF.        |
C|                                                          |
C------------------------------------------------------------
C
C     LOCATION OF THE 4 COEFFICIENTS
C                   |
C     RCOEF(1)      |  RCOEF(2)
C      -------------+------------
C     RCOEF(3)      |  RCOEF(4)
C                   |
C
C
C     SET UP THE REGRESSION PROBLEM
C     USE EVERY OTHER VALUE FOR THE PREDICTION PROCESS
C
      K = 0
      DO 80 J=1,16,2
      M = J + IXOFF
      DO 80 I=1,16,2
      N = I + IYOFF
      K = K + 1
      DO 70 L=1,4
   70 X(K,L) = D(N+QOFF(MIN,L,1),M+QOFF(MIN,L,2))
   80 OUT(K) = C(I,J)
C
C     GENERATE THE PREDICTION COEFFICIENTS FROM THE MODEL
C
      CALL COEFGN(X,OUT,RCOEF,K)
C
C     QUANTIZE PREDICTOR COEFFICIENTS TO NPBITS BITS
C
      DO 90 I=1,4
   90 RCOEF(I) = (AINT(RCOEF(I)*RLEVEL + .5))/RLEVEL
      IF(K.NE.129) GO TO 110
C
C     IF INVERSE DOESN'T EXIST, THEN SET COEFFICIENTS
C
  100 CONTINUE
C
C     INTEGER DISPLACEMENT ASSUMED AT THIS POINT
C     SET UP THE PREDICTOR COEFFICIENT VECTOR
C
      INTBIT = 0
      RCOEF(1) = 0.0
      RCOEF(2) = 0.0
      RCOEF(3) = 1.0
      RCOEF(4) = 0.0
      MIN = 1
C
C     CHECK IF WITH ERROR - NO DISPLACEMENT
C
      IF(IXOFF.EQ.10 .AND. IYOFF.EQ.10) GO TO 110
      CALL FILL(IBLK,JBLK,0,0,16,MTAB,24,16,'II')
      NBITS = NBITS + 18
  110 CONTINUE
C
C------------------------------------------------------------
C|                                                          |
C|     CALCULATE THE PREDICTED IMAGE BLOCK                  |
C|                                                          |
C------------------------------------------------------------
C
     CALL PRED(C,D,ERROR,36,16,IXOFF,IYOFF,VA,AV,QOFF,MIN,RCOEF,INTBIT,
    # NBITS,RETCOD,ER16,DIST,LVLARA)
      STD = VA
```

```
C|      START THE A IMAGE BLOCK TRANSFER INTO THE D SUB-    |
C|       MATRIX, THAT IS THE PAST FRAME MATRIX.             |
C|                                                           |
C------------------------------------------------------------
C
      ISTX = ISTX - 10
      ISPX = ISTX + 35
      ITY = ISTY - 10
      IPY = ITY + 35
      ISX = ISTX
      IHX = ISPX
      ISY = ITY
      IHY = IPY
      IF(IBLOCK.EQ.0) GO TO 50
      IF(ISX.LT.1) ISX = 1
      IF(IHX.GT.JSIZE) IHX = JSIZE
      IF(ISY.LT.1) ISY = 1
      IF(IHY.GT.ISIZE) IHY = ISIZE
   50 M = 0
      N = 0
      IF(ISTX.LT.1) M = 1 - ISTX
      IF(ITY.LT.1) N = 1 - ITY
      KCON(1) = N + 1
      KCON(2) = M + 1
      K = M
      DO 60 J=ISX,IHX
      K = K + 1
      L = N
      DO 60 I=ISY,IHY
      L = L + 1
   60 D(L,K) = IA(I,J)
      KCON(3) = L
      KCON(4) = K
C
C------------------------------------------------------------
C|                                                           |
C|        SOLVE FOR THE METRIC MATRIX-AND INTEGER DISP.      |
C|        THE ACTUAL METRIC VALUES ARE CONTAINED IN THE      |
C|        F MATRIX.                                          |
C|                                                           |
C------------------------------------------------------------
C
C     IF(IBLOCK.EQ.1) CALL INITI(F,21,21,0)
      IF(JPLT.EQ.1)CALL INITI(F,21,21,256)
      CALL METRIC(C,D,F,36,21,16,IMIN,LOC,IBLOCK,KCON,ITHRSH,11,11)
C
C     TEST IF ONLY INTEGER DISPLACEMENT - IF SO
C     GO TO IMAGE PREDICTION SECTION.
C
      IXOFF = LOC(2) - 1
      IYOFF = LOC(1) - 1
C     CALL MPLOT(F,XE,21,IPLT,IMAGE,IBLK,JBLK,LOC,NGRD)
      IF(IMIN.LE.ITHRSH) GO TO 100
      INTBIT = 0
C------------------------------------------------------------
C|                                                           |
C|    NON-INTEGER DISPLACEMENT ASSUMED AT THIS POINT         |
C|                                                           |
C------------------------------------------------------------
C
      NBITS =  NBITS + 18 + 4*NPBITS
      INTBIT = 1
C
C      DETERMINE IF PLOTS REQUIRED
C
C     IF(JPLT.EQ.1 .AND. IBLOCK.EQ.0 .AND. IMAGE.GT.5)
C    # CALL MPLOT(F,XE,21,IPLT,IMAGE,IBLK,JBLK,LOC,NGRD)
C
C------------------------------------------------------------
C|                                                           |
C|     SECTION FOR NON-INTEGER PORTION OF THE DISPLACE-      |
C|     MENT ESTIMATION.                                      |
C|                                                           |
C------------------------------------------------------------
C
```

```
C-----------------------------------------------------------------
C
      DO 260 IBLK = 1,ISTOP
      ISTY = (IBLK-1)*16 + 1
      ISPY = ISTY + 15
      DO 260 JBLK = 1,JSTOP
C
C-----------------------------------------------------------------
C|                                                               |
C|        COPY THE REQUIRED BLOCK FROM IMAGE B INTO              |
C|        MATRIX C THE PRESENT FRAME MATRIX.                     |
C|                                                               |
C-----------------------------------------------------------------
C*****************************************************************
C*        ISTX IS THE X STARTING POINTER FOR A AND B             *
C*        ISPX IS THE X ENDING   POINTER FOR A AND B             *
C*        ISTY IS THE Y STARTING POINTER FOR A AND B             *
C*        ISPY IS THE Y ENDING   POINTER FOR A AND B             *
C*****************************************************************
C
      ISTX = (JBLK-1)*16 + 1
      IX = ISTX
      ISPX = ISTX + 15
C
C     RESET INTBIT
C     INTBIT: 0 - INTEGER ONLY DISPLACEMENT
C             1 - NON-INTEGER DISPLACEMENT
C
      INTBIT = 0
C
C     TEST IF BLOCK IS ON THE BORDER, IF SO SET D MATRIX TO THE
C     AVERAGE OF THE B SUBMATRIX.  IBLOCK IS A BORDER FLAG
C     IF IBLOCK = 0 THEN NON-BORDER BLOCK
C     IF IBLOCK = 1 THEN BORDER BLOCK
C
      IF(IBLK .EQ. 1 .OR. IBLK .EQ. ISTOP) GO TO 20
      IF(JBLK .EQ. 1 .OR. JBLK .EQ. JSTOP) GO TO 20
C
C-----------------------------------------------------------------
C|                                                               |
C|      SECTION FOR NON-BORDER BLOCKS                            |
C|                                                               |
C-----------------------------------------------------------------
      IBLOCK = 0
      K = 0
      DO 10 J=ISTX,ISPX
      K = K + 1
      L = 0
      DO 10 I=ISTY,ISPY
      L = L + 1
   10 C(L,K) = B(I,J)
      GO TO 40
C
C-----------------------------------------------------------------
C|                                                               |
C|      SECTION FOR BORDER BLOCKS                                |
C|                                                               |
C-----------------------------------------------------------------
C
   20 IBLOCK = 1
      K = 0
      ITOTAL = 0
      DO 30 J=ISTX,ISPX
      K = K + 1
      L = 0
      DO 30 I=ISTY,ISPY
      L = L + 1
      ITOTAL = ITOTAL + IA(I,J)
   30 C(L,K) = B(I,J)
      IAVG = IFIX(FLOAT(ITOTAL)/256. + .5)
      CALL INITI(D,36,36,IAVG)
   40 CONTINUE
C
C-----------------------------------------------------------------
C|                                                               |
C       NTBITS - RUNNING TOTAL FOR BITS USED
C       NIMAGE - NUMBER OF IMAGES TO BE PROCESSED
C       SNRSET - MINIMUM OUTPUT SNR
C
        NPBITS = 10
        VAREST = 1.00
        STDERR = 3.0
        AVGERR = 3.0
        ISIZE = 192
        JSIZE = 128
        IPRT = 1
        IPLT = 0
        JPLT = 0
        KPLT = 0
        NTBITS = 0
        NIMAGE = 40
        SNRSET = 24.
C
C       END OF PROGRAM CONSTANT INITIALIZATION
C
        RLEVEL = FLOAT(2**NPBITS)
        NPBITS = NPBITS + 1
        ITHRS8 = VAREST*128
        ITHRSH = INT(VAREST*512. + .5)
        ISTOP = ISIZE/16
        JSTOP = JSIZE/16
        SIZE = FLOAT(ISIZE*JSIZE)
        SSST = 0.0
        SSET = 0.0
        SSPT = 0.0
        TART = 0.0
        DRTT = 0.0
        CALL READ(IA,ISIZE,JSIZE,21)
C
C       ALLOW FOR MORE THAN 1 ERROR
C
C       CALL ERRSET(208,1000,1,0,0)
C       CALL ERRSET(253,1000,1,0,0)
C
C       INITIALIZE HISTOGRAM VECTOR
C
        CALL INITR(HIST,1,258,0.0)
        CALL INITI(LVLARA,1,128,0)
        CALL QUANTI(DIST)
        IF(IPLT.EQ.1 .AND. KPLT.EQ.1)CALL PLOTS(IBUF,1,15)
C*********************************************************
C*                                                       *
C*       START ACTUAL IMAGE SEQUENCE LOOP                *
C*                                                       *
C*********************************************************
        IF(IPRT.GE.1)WRITE(6,1020)NPBITS,VAREST,STDERR,AVGERR,SNRSET
        DO 500 IMAGE = 1,NIMAGE
        CALL READ(B,ISIZE,JSIZE,21)
        IF(IPRT.GE.1)WRITE(6,1010)FF,IMAGE
        IF(IPRT.GE.2)WRITE(6,1030)
        IF(IPRT.GE.2)WRITE(6,1040)
C--------------------------------------------------------
C|                                                      |
C|       INITIALIZE IMAGE CONSTANTS                     |
C|                                                      |
C--- ---------------------------------------------------
C
C       NBITS - A COUNTER FOR THE DATA RATE
        NBITS = 0
C
C       INITIALIZE MOTION TAB ARRAY
C
        CALL INITI2(MTAB,24,16,16448)
C
C--------------------------------------------------------
C|                                                      |
C|           SET UP THE BLOCK COUNTERS TO STEP THROUGH THE
C|           IMAGE IN 16 BY 16 BLOCKS.  IBLK IS THE BLOCK
C|           ROW NUMBER AND JBLK IS THE BLOCK COLUMN NUMBER.|
C|
```

PROGRAM PCEC

```
C***********************************************************
C*                                                         *
C*       PROGRAM BY:  CARL BOWLING                         *
C*       LATEST UPDATE:  JULY 18,1983                      *
C*                                                         *
C*     PROGRAM FOR 2-D DELTA FUNCTION LOCATION DETERMIN-   *
C*        ATION.  TO BE USED FOR MOTION COMPENSATED IMAGE*
C*        CODING.                                          *
C*                                                         *
C*       SUBROUTINES NEEDED:                               *
C*                                                         *
C*       READ   - READS IN IMAGES                          *
C*       INITR  - INITIALIZES A REAL ARRAY TO A CONSTANT   *
C*       INITI  - INITIALIZES AN INTEGER ARRAY TO A CONST*
C*       INITI2 - INITIALIZES AN INTEGER*2 ARRAY TO A      *
C*                CONSTANT                                 *
C*       LOCATE - LOCATES THE MINIMUM QUADRANT VALUE       *
C*       METRIC - COMPUTES THE DIFFERENCE METRIC           *
C*       PRED   - CALCULATES THE PREDICTED VALUES          *
C*       COEFGN - PREDICTION COEFFICIENT GENERATOR         *
C*       COMTAL - OUTPUTS AN ARRAY TO TAPE IN COMTAL FORM*
C*       PLOT   - ALL THE CALCOMP DRIVER SOFTWARE AND      *
C*                THE TRANSLATOR SOFTWARE FOR THE TEK      *
C*                4662 PLOTTER.                            *
C*       INVERT - SCALES DATA 0-1 IN INVERTED ORDER        *
C*       PUR    - DRIVER FOR PURJOY 3D PLOT ROUTINE        *
C*       FILL   - KEEPS TRACK OF BLOCKS WITH MOTION        *
C*                                                         *
C***********************************************************
C
       INTEGER*2 IA(192,128),B(192,128),EST(192,128),MTAB(24,16),FF
       INTEGER*2 ERRSIG(192,128)
       INTEGER QOFF(4,4,2),KCON(4),C8(8,8),D8(28,28),E8(8,8),KCON8(4)
       INTEGER F(21,21),C(16,16),D(36,36),ERROR(16,16),LVLARA(128)
       INTEGER ER8(8,8),ER16(16,16),LOC(4),RETCOD
       REAL X(64,4),OUT(64),HIST(258),XHIST(258),RCOEF(4),XE(21,21)
       REAL XY(2,6),NGRD(4),SNRARY(50),SNRPAR(50),DRTARY(50),TARARY(50)
       REAL DIST(128)
       COMMON/THRESH/ STDERR,AVGERR,SNRSET
       COMMON/RCODE/ RETCOD
       DATA QOFF/2*0,2*-1,2*1,4*0,2*-1,2*1,2*0,-1,2*0,2*-1,2*0,-1,0,2*1,
     A           2*0,2*1,0/
       DATA FF/3084/
C***********************************************************
C*                                                         *
C*      PROGRAM CONSTANT INITIALIZATION SECTION            *
C*                                                         *
C***********************************************************
C
C      PROGRAM CONSTANT KEY
C
C      NPBITS - NUMBER OF BITS TO QUANTIZE PREDICTOR COEFFICIENTS
C               (ACTUAL NUMBER IS 1 GREATER TO INCLUDE SIGN)
C      VAREST - ESTIMATE FOR AVERAGE ABSOLUTE DIFFERENCE BETWEEN
C               CONSTANT FRAMES
C      STDERR - ERROR STANDARD DEVIATION THRESHOLD
C      AVGERR - ERROR AVERAGE THRESHOLD
C      ISIZE  - INPUT HEIGHT OF THE PICTURE IN PIXELS
C      JSIZE  - INPUT WIDTH OF THE PICTURE IN PIXELS
C      IPRT   - PRINTER FLAG
C             = 0: DON'T PRINT DATA
C             = 1: PRINT DATA
C             = 2: PRINT ALL DATA (STEP BY STEP)
C      IPLT   - PLOTTER DIRECTOR FLAG
C             = 0: PLOT WILL BE DIRECTED TO 4662 PLOTTER VIA GPIB
C             = 1: PLOT WILL BE DIRECTED TO THE 4025 SCREEN
C               NOTE: MAKE SURE THE REQUIRED TXTLIBS ARE AVAILABLE
C      JPLT   - METRIC PLOT FLAG
C             = 0: NO METRIC PLOT DRAWN
C             = 1: METRIC PLOT WILL BE DRAWN
C      KPLT   - PLOTTER FLAG
C             = 0: NO PLOTTING IS NEEDED
C             = 1: SOME TYPE OF PLOTTING IS REQUIRED
```

Appendix B

MOTION COMPENSATED IMAGE CODING USING PREDICTION COEFFICIENT ENERGY CONCENTRATION - PROGRAM LISTINGS

SUBROUTINE XFORM

```
      SUBROUTINE XFORM(C,V,XI,H,N,M)
C**************************************************************
C*                                                            *
C*    SUBROUTINE TO PERFORM A LINEAR TRANSFORMATION           *
C*                                                            *
C*    TRANSFORMATION IS GIVEN BY:                             *
C*                               (V)(II)(H)                   *
C*                                                            *
C**************************************************************
C
      REAL C(N,M),V(2,2),H(M,M),T(6,8),XI(N,M)
C
C     CALCULATE (V)*(XI)
C
      DO 30 L=1,6,2
      I1=L
      I2=L+1
      DO 10 I=I1,I2
      NN = I-L+1
      DO 10  J=1,8
      T(I,J)=0.0
      DO 10 K=1,2
      I3 = L + K - 1
      T(I,J) = T(I,J) + V(NN,K)*XI(I3,J)
   10 CONTINUE
C
C     CALCULATE ((V)(XI))*H
C
      DO 20 I=I1,I2
      DO 20 J=1,8
      C(I,J)=0.0
      DO 20 K=1,8
      C(I,J)=C(I,J)+T(I,K)*H(K,J)
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
```

SUBROUTINE RINTRP

```
      FUNCTION RINTRP(I1,I2,X)
C********************************************************
C*                                                      *
C*    FUNCTION RINTRP                                   *
C*                                                      *
C*    PURPOSE:                                          *
C*           THIS FUNCTION IS USE TO PERFORM A LINEAR   *
C*           INTERPOLATION BETWEEN INPUT INTEGER POINTS *
C*                                                      *
C*    EXPLANATION OF CALL VARIABLES                     *
C*           I1 - LOWER BOUND VALUE                     *
C*           I2 - UPPER BOUND VALUE                     *
C*           X  - THE NON-INTEGER INTERPOLATION         *
C*                                                      *
C********************************************************
C
      Y1 = FLOAT(I1)
      Y2 = FLOAT(I2)
      IF(X.EQ.0.0) RI = Y1
      IF(X.EQ.0.0) RETURN
      RINTRP = Y1 + (Y2 - Y1)*X
      RETURN
      END
```

```
   50 S2 = S2 + R(K)*V1(K)
      DO 60 K=1,N
      G(K) = V1(K)/(VV + S2)
   60 RT = RT + A(K)*R(K)
      DO 70 K=1,N
      DO 70 L=1,K
      V(K,L) = V(K,L) - G(K)*V1(L)
   70 V(L,K) = V(K,L)
      E = S - RT
      DO 80 K=1,N
   80 A(K) = A(K) + G(K)*E
C
C     SHIFT THE PAST VALUE VECTOR
C
      DO 90 L=2,N
   90 R(N+2-L) = R(N+1-L)
      R(1) = S
  100 CONTINUE
      DO 110 K=1,N
  110 CM(K,I) = A(K)
C
C     RESET R-REGISTER TO INITIAL SIGNAL VALUE
C
      DO 120 J=1,N
  120 R(J) = X1
      DO 150 J=1,NSF
      S = XV(J)
      RT = 0.0
C
C     DETERMINE RESIDUAL SIGNAL
C
      DO 130 K=1,N
 130  RT = RT + A(K)*R(K)
      E=S-RT
      XIN(I,J) = E
C
C     SHIFT R-REGISTER
C
      DO 140 K=2,N
  140 R(N+2-K) = R(N+1-K)
      R(1) = S
  150 CONTINUE
  160 CONTINUE
      RETURN
      END
```

SUBROUTINE COMP

```
      SUBROUTINE COMP(XIN,NF,NSF,N,CM)
C
C***********************************************************
C*                                                         *
C*    SUBROUTINE - COMP                                    *
C*    PURPOSE:                                             *
C*              THIS SUBROUTINE IS USED TO PERFORM ADAPTIVE*
C*              HYBRID PICTURE CODING (AHPC) ON AN INPUT   *
C*              ARRAY.                                     *
C*                                                         *
C*        EXPLANATION OF CALL VARIABLES:                   *
C*    XIN     - THE INPUT DATA AND RESIDUAL OUTPUT MATRIX  *
C*    NF      - NUMBER OF LINES IN THE INPUT MATRIX        *
C*    NSF     - NUMBER OF SAMPLES PER FRAME-SAMPLES/LINE   *
C*    N       - NUMBER OF PREDICTOR COEFFICIENTS TO BE USED*
C*    CM      - MATRIX CONTAINING PREDICTOR COEFFICIENTS   *
C*                                                         *
C*        DEFINITION OF VARIABLE TERMS:                    *
C*    V       - THE ERROR COVARIANCE MATRIX                *
C*    A       - THE PREDICTOR COEFFICIENT VECTOR           *
C*    VARI    - INITIAL VALUE FOR ERROR COVARIANCE MATRIX  *
C*    VV      - VARIANCE OFFSET                            *
C*    XV      - THE INPUT LINE TEMPORARY VECTOR            *
C*    G       - THE GAIN VECTOR                            *
C*    R       - THE PAST VALUE VECTOR                      *
C*    E       - THE ERROR OR RESIDUAL TERM                 *
C*                                                         *
C*        EXTERNAL ROUTINES REQUIRED:                      *
C*    NONE                                                 *
C*                                                         *
C***********************************************************
C
      REAL A(6),V1(6),XIN(NF,NSF),R(6),XV(256),G(6),V(6,6),CM(N,NF)
C
C     SET UP THE CONSTANTS AND INITIAL VALUES
C
      DATA V/36*0.0/,A/1.0,-.5,-.2,.3,.4,-.5/
      XNSF = NSF
      VV = 1.0
      VARI = 100.0
C
C     IN THE DO 160 LOOP, I IS THE LINE NUMBER (1 - NF)
C
      DO  160 I=1,NF
      DO 10 J=1,N
      DO 10 K=1,N
      V(J,K) = 0.0
      IF(J.EQ.K) V(J,K) = VARI
   10 CONTINUE
C
C     X1 IS THE FIRST VALUE OF EACH LINE, USED TO INITIALIZE THE R VECT
C
      X1 = XIN(I,1)
C
C      SET UP THE INPUT VECTOR AND THE PAST VALUE VECTOR
C
      DO 20 J=1,NSF
   20 XV(J) = XIN(I,J)
      DO 30 J=1,N
   30 R(J) = X1
C
C      IDENTIFICATION LOOP (IDENTIFY THE PREDICTOR COEFFICIENTS)
C
C
C
      DO 100 J=1,NSF
      S = XV(J)
      S2 = 0.0
      RT = 0.0
      DO 40 K=1,N
      V1(K) = 0.
      DO 40 L=1,N
   40 V1(K) = V1(K) + V(K,L)*R(L)
      DO 50 K=1,N
```

```
       IF(MTAB(I,K).NE.16448)NTAB = NTAB + 1
  325  CONTINUE
       IF(IPRT.GE.1)WRITE(6,1110)(MTAB(I,K),K=1,16)
  330  CONTINUE
       TARARY(IMAGE) = NTAB
       TART = TART + NTAB
      IF(IPRT.GE.1)WRITE(6,1100)
       DRATE = FLOAT(NBITS)/SIZE
       DRTT = DRTT + DRATE
       SSST = SSST + SSS
       SSET = SSET + SSE
       SSPT = SSPT + SSP
       DRTARY(IMAGE) = DRATE
       NTBITS = NTBITS + NBITS
       TDRATE = FLOAT(NTBITS)/(FLOAT(IMAGE)*SIZE)
       IF(IPRT.GE.1)WRITE(6,1120)NBITS,DRATE,NTBITS,TDRATE
  500  CONTINUE
       SNR = 10.*ALOG10(SSST/SSET)
       SNRP = 10.*ALOG10(SSST/SSPT)
       SNRARY(NIMAGE+1) = SNR
       SNRPAR(NIMAGE+1) = SNRP
       DRTARY(NIMAGE+1) = DRTT/FLOAT(NIMAGE)
       TARARY(NIMAGE+1) = TART/FLOAT(NIMAGE)
       NN = NIMAGE + 1
       WRITE(9)NN
       WRITE(9)(SNRARY(I),I=1,NN)
       WRITE(9)(SNRPAR(I),I=1,NN)
       WRITE(9)(DRTARY(I),I=1,NN)
       WRITE(9)(TARARY(I),I=1,NN)
       WRITE(9)(HIST(I),I=1,256)
       WRITE(9)(LVLARA(I),I=1,128)
       IF(IPRT.GE.1)WRITE(6,1080)SNR,SNRP
 1010  FORMAT(1X,A1,'IMAGE SEQUENCE NUMBER ',I2)
 1020  FORMAT(1X,'BITS/COEF. = ',I2,' VAREST = ',F5.2,
     # ' STDERR = ',F5.2,' AVGERR = ',F5.2,' SNRSET = ',F5.2)
 1030  FORMAT(/,'      Y     X  QUAD XDIS YDIS    PRE-AVG    PRE-STD    AFT-A
     #VG    AFT-STD',/)
 1040  FORMAT(1X,80('-'))
 1050  FORMAT(2X,2I5,I4,2I5,2F10.3,3I7)
 1060  FORMAT(1X,5I5,4F10.3,2I7)
 1070  FORMAT(1X,2F10.1)
 1080  FORMAT(1X,'SIGNAL TO NOISE RATIO ',2F10.5,' DB')
 1090  FORMAT(1X,A1)
 1100  FORMAT(1X,'_____________________________________________')
 1110  FORMAT(1X,'|',16(1X,A1),'|')
 1120  FORMAT(1X,'#BITS =',I7,' .  DATA RATE  ',F6.4,' TOTAL = ',I7,
     # ' CUMULATIVE DATA RATE ',F6.4)
       STOP
       END
       BLOCK DATA
C
C      BLOCK DATA FOR POINTERS FOR SUBROUTINE METRIC
C      IN ORDER THEY ARE XSTART,XSTOP,YSTART,YSTOP
C      MAX IS 21 AND MIN IS 1
C
       COMMON /PNTR/IRANGE
       INTEGER IRANGE(4)/8,14,8,14/
       END
```

SUBROUTINE CHANGE

```
      SUBROUTINE CHANGE(A,B,NBY,N,IDIR)
C************************************************************
C*                                                          *
C*      SUBROUTINE:CHANGE                                   *
C*      PURPOSE:                                            *
C*              THIS SUBROUTINE IS USED TO CONVERT A REAL*4*
C*              INTO A REAL*8 AND VIS-VERSA ACCORDING TO   *
C*              THE VALUE OF IDIR                           *
C*                                                          *
C*      EXPLANATION OF CALL VARIABLES:                      *
C*              A       - B - REAL*4 ARRAY                  *
C*              B       - B - REAL*8 ARRAY                  *
C*              NBY     - I - IST SIZE VARIABLE FOR A AND B *
C*              N       - I - 2ND SIZE VARIABLE FOR A AND B *
C*              IDIR    - I - DIRECTION FLAG FOR TRANSFER   *
C*                            +1  MOVE A -> B               *
C*                            -1  MOVE B -> A               *
C*                                                          *
C*      SUBROUTINES NEEDED:                                 *
C*              NONE                                        *
C*                                                          *
C************************************************************
      REAL A(NBY,N)
      REAL*8 B(NBY,N)
      IF(IDIR.LT.0) GO TO 20
      DO 10 J=1,N
      DO 10 I=1,NBY
   10 B(I,J) = A(I,J)
      RETURN
   20 DO 30 J=1,N
      DO 30 I=1,NBY
   30 A(I,J) = B(I,J)
      RETURN
      END
```

SUBROUTINE COEFGN

```
      SUBROUTINE COEFGN(X,OUT,COEF,IERN)
C**************************************************************
C*                                                            *
C*     SUBROUTINE:COEFGN                                      *
C*     PURPOSE:                                               *
C*             THIS SUBROUTINE IS USED TO COMPUTE A SET OF*
C*             REGRESSION COEFFICIENTS TO FIT THE DATA        *
C*             INPUT IN THE MATRIX X AND THE VECTOR OUT.      *
C*             THE COEFFICIENTS ARE THEN OUTPUT IN THE        *
C*             RCOEF VECTOR OR THE COEF VECTOR.               *
C*                                                            *
C*     EXPLANATION OF CALL VARIABLES:                         *
C*             X       - I - MATRIX OF SIZE 4 BY 64 WHICH     *
C*                           CONTAINS THE REGRESSION DATA     *
C*             OUT     - I - VECTOR OF SIZE 64 CONTAINING     *
C*                           THE REGRESSION DATA              *
C*             RCOEF   - 0 - OUTPUT VECTOR OF SIZE 4 THAT     *
C*                           CONTAINS THE INEQUALITY          *
C*                           CORRECTED REGRESSION             *
C*                           COEFFICIENTS.                    *
C*             COEF    - 0 - OUTPUT VECTOR OF SIZE 4 THAT     *
C*                           CONTAINS THE REGULAR             *
C*                           REGRESSION COEFFICIENTS.         *
C*             IERN    - 0 - ERROR CHECK FOR NON-EXISTANCE *
C*                           OF INVERSE                       *
C*                                                            *
C*     NOTE: ONLY 1 OF THE COEFFICIENT GENERATION METHODS *
C*           CAN BE USED.  THE CHOICE IS MADE BY PLACING      *
C*           THE NAME RCOEF OR COEF IN THE SUBROUTINE         *
C*           STATEMENT - THIS VECTOR IS THEN RETURNED TO      *
C*           THE CALLING PROGRAM.                             *
C*                                                            *
C*     EXPLANATION OF CALL VARIABLES:                         *
C*     SUBROUTINES NEEDED:                                    *
C*             TRANSP - TRANSPOSES A MATRIX                   *
C*             MMUL   - MATRIX MULTIPLICATION ROUTINE         *
C*             CHANGE - CHANGES REAL*8 TO REAL*4 AND BACK *
C*             LINV2F - MATRIX INVERSE (IMSL ROUTINE)         *
C*                                                            *
C**************************************************************
      REAL X(64,4),XT(4,64),XTX(4,4),XTXI(4,4),COEF(4),OUT(64),XTOUT(4)
      REAL A(4),XTXIA(4),RCOEF(4),XTXIXT(4,64)
      REAL*8 XTX8(4,4),XTXI8(4,4),WORK(100)
      DATA A/4*1./
      CALL TRANSP(X,XT,64,4,64,4)
      CALL MMUL(XT,X,XTX,4,64,4,4,64,4)
      CALL CHANGE(XTX,XTX8,4,4,1)
      CALL LINV2F(XTX8,4,4,XTXI8,3,WORK,IERN)
C
C     CHECK FOR PROBLEMS WITH THE INVERSE EXISTING, IF PROBLEM
C     DOES EXIST RETURN TO MAIN AND CORRECT
C
      IF(IERN.EQ.129)RETURN
      CALL CHANGE(XTXI,XTXI8,4,4,-1)
      CALL MMUL(XT,OUT,XTOUT,4,64,1,4,64,1)
      CALL MMUL(XTXI,XTOUT,COEF,4,4,1,4,4,1)
C
C     ADD CORRECTION FOR EQUALITY AND INEQUALITY CONSTRAINTS
C
      CTOTAL = 0.0
      TOTAL  = 0.0
      DO 10 I=1,4
      CTOTAL = CTOTAL + COEF(I)
      DO 10 J=1,4
   10 TOTAL = TOTAL + XTXI(I,J)
      TOTAL = 1./TOTAL
      CTOTAL = 1. - CTOTAL
      CALL MMUL(XTXI,A,XTXIA,4,4,1,4,4,1)
      TOTAL = TOTAL*CTOTAL
      DO 20 I=1,4
   20 RCOEF(I) = XTXIA(I)*TOTAL + COEF(I)
      RETURN
      END
```

SUBROUTINE COMTAL

```
      SUBROUTINE COMTAL(IN,NBY,N,LOUT)
C***********************************************************
C*                                                         *
C*      SUBROUTINE:COMTAL                                  *
C*      PURPOSE:                                           *
C*              THIS SUBROUTINE IS USED TO TRANSLATE AN    *
C*              INTEGER*2 ARRAY TO 1 BYTE PIXEL VALUES AND *
C*              WRITE IT OUT IN COMTAL FORMAT TO DISK OR   *
C*              TAPE                                       *
C*      EXPLANATION OF CALL VARIABLES:                     *
C*              IN      - I - INTEGER*2 IMAGE ARRAY        *
C*              NBY     - I - 1ST SIZE VARIABLE FOR IN     *
C*              N       - I - 2ND SIZE VARIABLE FOR IN     *
C*              LOUT    - I - LOGICAL OUTPUT NUMBER FOR THE *
C*                            DATA TO BE WRITTEN           *
C*                                                         *
C*      SUBROUTINES NEEDED:                                *
C*              NONE                                       *
C*                                                         *
C***********************************************************
      INTEGER*2 IN(NBY,N),IWORK(512)
      LOGICAL*1 OUT(2,512)
      EQUIVALENCE (IWORK(1),OUT(1,1))
      DO 20 I=1,NBY
      DO 10 J=1,N
   10 IWORK(J) = IN(I,J)
   20 WRITE(LOUT,30)(OUT(2,J),J=1,N)
   30 FORMAT(4(128A1))
      RETURN
      END
```

SUBROUTINE FILL

```
      SUBROUTINE FILL(IBLK,JBLK,IBLK8,JBLK8,IBLKSZ,MDATA,NBY,N,KIND)
C***********************************************************
C*                                                         *
C*      SUBROUTINE:FILL                                    *
C*      PURPOSE:                                           *
C*              THIS SUBROUTINE IS USED TO KEEP AN OUTPUT  *
C*              ARRAY WITH THE TYPE OF MOTION INVOLVED.    *
C*              AN 'I' IN THE OUTPUT ARRAY MEANS THAT THE  *
C*              MOTION FOR THAT BLOCK OR SUB-BLOCK WAS     *
C*              INTEGER ONLY.  AN 'R' IMPLIES THAN SOME    *
C*              NON-INTEGER PORTION OF THE DISPLACEMENT WAS*
C*              USED                                       *
C*                                                         *
C*      EXPLANATION OF CALL VARIABLES:                     *
C*              IBLK   - I - Y LOCATION OF MAJOR BLOCK     *
C*              JBLK   - I - X LOCATION OF MAJOR BLOCK     *
C*              IBLK8  - I - Y LOCATION O SUB-BLOCK        *
C*              JBLK8  - I - X LOCATION OF SUB-BLOCK       *
C*              IBLKSZ - I - CURRENT BLOCKSIZE             *
C*              MDATA  - O - ARRAY FOR INFORMATION         *
C*              NBY    - I - 1ST SIZE VARIABLE FOR MDATA   *
C*              N      - I - 2ND SIZE VARIABLE FOR MDATA   *
C*              KIND   - I - MOTION TYPE IN QUOTES:        *
C*                           'RR' -> NON-INTEGER DISP.     *
C*                           'II' -> INTEGER DISPLACEMENT  *
C*                           '  ' -> NO DISPLACEMENT       *
C*                                                         *
C*      SUBROUTINES NEEDED:                                *
C*              NONE                                       *
C*                                                         *
C***********************************************************
      INTEGER*2 MDATA(NBY,N),KIND
      MX1 = 2*(IBLK-1) + 1
      MX2 = MX1 + 1
      MY1 = 2*(JBLK-1) + 1
      MY2 = MY1 + 1
      IF(IBLKSZ.EQ.8) GO TO 10
      MDATA(MX1,MY1) = KIND
      MDATA(MX1,MY2) = KIND
      MDATA(MX2,MY1) = KIND
      MDATA(MX2,MY2) = KIND
      RETURN
  10  CONTINUE
      MX2 = MX1 + IBLK8
      MY2 = MY1 + JBLK8
      MDATA(MX2,MY2) = KIND
      RETURN
      END
```

SUBROUTINE INITI

```
      SUBROUTINE INITI(A,N1,N2,IVAL)
C********************************************************
C*                                                      *
C*     SUBROUTINE:INITI                                 *
C*     PURPOSE:                                         *
C*            THIS SUBROUTINE IS USED TO INITIALIZE A   *
C*            INTEGER ARRAY TO A CONSTANT VALUE         *
C*                                                      *
C*     EXPLANATION OF CALL VARIABLES:                   *
C*            A     - 0 - INTEGER ARRAY TO INITIALIZE   *
C*            N1    - I - 1ST SIZE VARIABLE FOR A       *
C*            N2    - I - 2ND SIZE VARIABLE FOR A       *
C*            IVAL  - I - VALUE THE ARRAY IS INITIALIZED*
C*                        TO                            *
C*     SUBROUTINES NEEDED:                              *
C*            NONE                                      *
C*                                                      *
C********************************************************
      INITIGER A(N1,N2)
      DO 10 J=1,N2
      DO 10 I=1,N1
   10 A(I,J) = IVAL
      RETURN
      END
```

SUBROUTINE INITI2

```
      SUBROUTINE INITI2(A,N1,N2,IVAL)
C************************************************************
C*                                                          *
C*      SUBROUTINE:INITI2                                   *
C*      PURPOSE:                                            *
C*              THIS SUBROUTINE IS USED TO INITIALIZE A     *
C*              INTEGER*2 ARRAY TO A CONSTANT VALUE         *
C*                                                          *
C*      EXPLANATION OF CALL VARIABLES:                      *
C*              A       - O - INTEGER ARRAY TO INITIALIZE   *
C*              N1      - I - 1ST SIZE VARIABLE FOR A       *
C*              N2      - I - 2ND SIZE VARIABLE FOR A       *
C*              IVAL    - I - VALUE THE ARRAY IS INITIALIZED*
C*                            TO                            *
C*      SUBROUTINES NEEDED:                                 *
C*              NONE                                        *
C*                                                          *
C************************************************************
      INTEGER*2 A(N1,N2)
      DO 10 J=1,N2
      DO 10 I=1,N1
   10 A(I,J) = IVAL
      RETURN
      END
```

SUBROUTINE INITR

```
      SUBROUTINE INITR(A,N1,N2,VAL)
C***********************************************************************
C*                                                                     *
C*      SUBROUTINE:INITR                                               *
C*      PURPOSE:                                                       *
C*              THIS SUBROUTINE IS USED TO INITIALIZE A                *
C*              REAL ARRAY TO A CONSTANT VALUE                         *
C*                                                                     *
C*      EXPLANATION OF CALL VARIABLES:                                 *
C*              A       - O - REAL ARRAY TO BE INITIALIZED             *
C*              N1      - I - 1ST SIZE VARIABLE FOR A                  *
C*              N2      - I - 2ND SIZE VARIABLE FOR A                  *
C*              VAL     - I - VALUE THE ARRAY IS INITIALIZED*
C*                            TO                                       *
C*      SUBROUTINES NEEDED:                                            *
C*              NONE                                                   *
C*                                                                     *
C***********************************************************************
      REAL A(N1,N2)
      DO 10 J=1,N2
      DO 10 I=1,N1
   10 A(I,J) = VAL
      RETURN
      END
```

SUBROUTINE INVERT

```
      SUBROUTINE INVERT(IA,A,N)
C***********************************************************
C*                                                         *
C*      SUBROUTINE:INVERT                                  *
C*      PURPOSE:                                           *
C*              THIS SUBROUTINE IS USED TO INVERT THE SCALE*
C*              ORDER AND SCALE THE INPUT DATA TO BETWEEN  *
C*              0 AND 1                                    *
C*      EXPLANATION OF CALL VARIABLES:                     *
C*              IA      - I - INTEGER ARRAY OF INPUT DATA  *
C*              A       - O - OUTPUT SCALED REAL ARRAY     *
C*              N       - I - SIZE OF THE INPUT AND OUTPUT *
C*                            ARRAYS                       *
C*                                                         *
C*      SUBROUTINES NEEDED:                                *
C*              NONE                                       *
C*                                                         *
C***********************************************************
      DIMENSION IA(N,N),A(N,N)
C
C     FIND MAX AND MIN VALUES
C
      MAX = -1000000
      MIN =  1000000
      DO 10 J=1,N
      DO 10 I=1,N
      IF(IA(I,J).GT.MAX)MAX = IA(I,J)
      IF(IA(I,J).LT.MIN)MIN = IA(I,J)
   10 CONTINUE
      XMAX = MAX
      DIF = MAX - MIN
      WRITE(6,11)MAX,MIN,DIF
   11 FORMAT(1X,2I10,F10.4)
C
C     REORDER AND SCALE
C
      DO 20 J=1,N
      DO 20 I=1,N
      A(I,J) = (XMAX - FLOAT(IA(I,J)))/DIF
   20 CONTINUE
      RETURN
      END
```

SUBROUTINE LOCATE

```
      SUBROUTINE LOCATE(LOC,E,NBY,N,MIN)
C*************************************************************
C*                                                           *
C*     SUBROUTINE:LOCATE                                     *
C*     PURPOSE:                                              *
C*             THIS SUBROUTINE IS USED FOR LOCATION THE      *
C*             NON-INTEGER PORTION OF THE DISPLACEMENT       *
C*             VIA FINDING THE MINIMUM QUADRANT              *
C*                                                           *
C*     EXPLANATION OF CALL VARIABLES:                        *
C*             LOC     - I - METRIC MINIMUM LOCATION (X,Y)   *
C*             E       - I - INPUT MATRIX HOLDING METRIC     *
C*             NBY     - I - 1ST SIZE VARIABLE FOR E         *
C*             N       - I - 2ND SIZE VARIABLE FOR E         *
C*             MIN     - O - MINIMUM QUADRANT NUMBER         *
C*                                                           *
C*     SUBROUTINES NEEDED:                                   *
C*             NONE                                          *
C*                                                           *
C*************************************************************
C
      INTEGER Q(4),E(NBY,N),LOC(2)
      COMMON /PNTR/IRANGE(4)
C     LOCATE ABSOLUTE MINIMUM QUADRANT BY SUMMING CORNER
C     VALUES OF QUADRANTS AS GIVEN BELOW.
C
C          |
C      IV  |  I
C   -----------
C     III  |  II
C          |
C
C
C
C      TEST IF INTEGER DISPLACEMENT ESTIMATE AT BORDER OF METRIC
C      CALCULATIONS. (ERROR WILL RESULT IN THAT METRIC VALUES OUTSIDE
C      DISPLACEMENT STEPS WILL BE ZERO.)
C
      I = LOC(1)
      J = LOC(2)
      IM1 = I - 1
      IP1 = I + 1
      JM1 = J - 1
      JP1 = J + 1
      IF(I.EQ.IRANGE(1) .OR. I.EQ.IRANGE(2)) GO TO 20
      IF(J.EQ.IRANGE(3) .OR. J.EQ.IRANGE(4)) GO TO 40
C
C     DISPLACEMENT ESTIMATE NOT ON BORDER OF METRIC
C
      Q(1) = E(IM1,J) + E(IM1,JP1) + E(I,JP1)
      Q(2) = E(I,JP1) + E(IP1,JP1) + E(IP1,J)
      Q(3) = E(IP1,J) + E(IP1,JM1) + E(I,JM1)
      Q(4) = E(I,JM1) + E(IM1,JM1) + E(IM1,J)
C
C     FIND MIN QUADRANT
C
      IMIN = Q(1)
      MIN = 1
      DO 10 M=2,4
      IF(Q(M) .GE. IMIN) GO TO 10
      MIN = M
      IMIN = Q(M)
   10 CONTINUE
      RETURN
   20 CONTINUE
      IF(J.EQ.IRANGE(3).OR.J.EQ.IRANGE(4)) GO TO 60
C
C    SECTION FOR ESTIMATE ON BORDER IN THE Y DIRECTION
C
      IF(I.EQ.IRANGE(2)) GO TO 30
C
C     ESTIMATE LOW IN Y DIRECTION ONLY
C     TEST IF IN QUADRANT II OR III
C
```

```
      Q(2) = E(JP1,I) + E(IP1,JP1)
      Q(3) = E(I,JM1) + E(IP1,JM1)
      MIN = 2
      IF(Q(3).LT.Q(2))MIN=3
      RETURN
   30 CONTINUE
C
C     HIGH IN THE Y DIRECTION ONLY
C     TEST IF IN QUADRANT I OR IV ONLY
C
      Q(1) = E(IM1,JP1) + E(I,JP1)
      Q(4) = E(IM1,JM1) + E(I,JM1)
      MIN = 1
      IF(Q(4).LT.Q(1)) MIN = 4
      RETURN
   40 CONTINUE
C
C    SECTION FOR ESTIMATE ON BORDER IN THE X DIRECTION
C
      IF(J.EQ.IRANGE(4)) GO TO 50
C
C     ESTIMATE LOW IN THE X DIRECTION ONLY
C     TEST IF IN QUANDRANT I OR II ONLY
C
      Q(1) = E(IM1,J) + E(IM1,JP1)
      Q(2) = E(IP1,J) + E(IP1,JP1)
      MIN = 1
      IF(Q(2).LT.Q(1)) MIN = 2
      RETURN
   50 CONTINUE
C
C     HIGH IN THE X DIRECTION ONLY
C     TEST IF IN QUADRANT III OR IV ONLY
C
      Q(3) = E(IP1,JM1) + E(IP1,J)
      Q(4) = E(I-1,J-1) + E(I-1,J)
      MIN = 3
      IF(Q(4).LT.Q(3)) MIN = 4
      RETURN
   60 CONTINUE
C
C     SECTION FOR ESTIMATE ON ONE OF THE CORNERS
C     THIS WILL FIX THE QUADRANT SUCH THAT IT LIES ON THE INTERIOR
C     OF THE METRIC MATRIX.
C
      MIN = 1
      IF(I.EQ.IRANGE(2) .AND. J.EQ.IRANGE(4)) MIN = 2
      IF(I.EQ.IRANGE(1) .AND. J.EQ.IRANGE(4)) MIN = 3
      IF(I.EQ.IRANGE(1) .AND. J.EQ.IRANGE(2)) MIN = 4
      RETURN
      END
```

SUBROUTINE METRIC

```
      SUBROUTINE METRIC(A,B,D,N1,N2,N3,IMIN,LL,IB,KCON,ITHR,I1,J1)
C***********************************************************************
C*                                                                    *
C*          SUBROUTINE:METRIC                                         *
C*          PURPOSE:                                                  *
C*                  THIS SUBROUTINE IS USED TO CALCULATE SOME         *
C*                  PREDEFINED METRIC BETWEEN THE TWO INPUT           *
C*                  MATRICES A AND B AND OUTPUT THE VALUES OF         *
C*                  THE METRIC IN MATRIX D. (ALL INTEGER)             *
C*                                                                    *
C*          EXPLANATION OF CALL VARIABLES:                            *
C*                  A       - I - 1ST MATRIX FOR METRIC               *
C*                  B       - I - 2ND MATRIX FOR METRIC               *
C*                  D       - O - OUTPUT MATRIX CONTAINING THE        *
C*                                METRIC VALUES - METHOD OF           *
C*                                THRESHOLD EXCEEDING COUNTING        *
C*                  N1      - I - SIZE VARIABLE FOR B                 *
C*                  N2      - I - SIZE VARIABLE FOR D                 *
C*                  N3      - I - CURRENT IMAGE BLOCKSIZE             *
C*                  IMIN    - I - MINIMUM VALUE FOR SUM OF THE        *
C*                                ABSOLUTE VALUE OF THE ERROR         *
C*                                AT THE MINIMUM METRIC LOCATION*
C*                  LL(1)   - O - OUTPUT 1ST METRIC LOCATION          *
C*                  LL(2)   - O - OUTPUT 2ND METRIC LOCATION          *
C*                  IB      - I - BORDER BLOCK FLAG (0/1)             *
C*                  KCON    - I - START/STOP FLAG VECTOR              *
C*                  ITHR    - I - DIFFERENCE THRESHOLD FOR 1ST        *
C*                                TEST RETURN                         *
C*                  I1      - I - A PRIORI X INTEGER ESTIMATE         *
C*                                FOR THE DISPLACEMENT                *
C*                  J1      - I - A PRIORI Y INTEGER ESTIMATE         *
C*                                FOR THE DISPLACEMENT                *
C*                                                                    *
C*          SUBROUTINES NEEDED:                                       *
C*                  NONE                                              *
C*                                                                    *
C***********************************************************************
      INTEGER LL(4),D(N2,N2),A(N3,N3),B(N1,N1),KCON(4),IPOS(256,2)
      INTEGER*2 E(16,16)
      COMMON /PNTR/ IRANGE(4)
C     IMIN = 2000000
      KTHR = 3
      KTEST = N3 **2
      IISIZE = KTEST
      SIZE = IISIZE
C
C     SET UP INITIAL DISPLACEMENT ESTIMATE (A PRIORI GUESS)
C
      ISTART = I1
      ISTOP  = I1
      JSTART = J1
      JSTOP  = J1
      ICOUNT = 1
C
C     DETERMINE IF CURRENT BLOCK IS BORDER BLOCK
C
      IF(IB.EQ.1) GO TO 80
C
C*********************************************************************
C*                                                                   *
C*    SECTION FOR NON-BORDER BLOCKS                                  *
C*                                                                   *
C*********************************************************************
C
   10 CONTINUE
      KTEST = IISIZE
C
C     THE DO 30 LOOPS ADJUST THE DISPLACEMENT ESTIMATE
C
      DO 30 I=ISTART,ISTOP
      IM1 = I - 1
      DO 30 J=JSTART,JSTOP
C
```

```
C     KCOUNT IS USED FOR THE THRESHOLD EXCEEDING COUNTING METHOD
C     KSUM IS USED FOR SUM OF ABSOLUTE VALUE OF DIFFERENCE METHOD
C
      KCOUNT = 0
      KSUM = 0
      JM1 = J - 1
C
C     THE DO 20 LOOPS EVALUATE A SINGLE METRIC VALUE
C
      DO 20 K=1,N3
      IK = K + IM1
      DO 20 L = 1,N3
      JL = L + JM1
      KDIF = IABS(A(L,K) - B(JL,IK))
      KSUM = KSUM + KDIF
      IF(KDIF.GT.KTHR)KCOUNT = KCOUNT + 1
   20 CONTINUE
      D(J,I) = KCOUNT
      E(J,I) = KSUM
      IF(KCOUNT.GT.KTEST) GO TO 30
C
C     RESET COUNTING METHOD POINTERS AND MINIMUM VALUE
C
      KTEST = KCOUNT
      LL(1) = J
      LL(2) = I
   30 CONTINUE
C
C     TEST FOR POSSIBLE MULTIPLE MINIMA (NCOUNT = #MINIMUMS)
C
      NCOUNT = 0
      DO 40 I=ISTART,ISTOP
      DO 40 J=JSTART,JSTOP
      IF(D(I,J).NE.KTEST)GO TO 40
      NCOUNT = NCOUNT + 1
      IPOS(NCOUNT,1) = I
      IPOS(NCOUNT,2) = J
   40 CONTINUE
      IF(NCOUNT.EQ.1) GO TO 70
C
C     MULTIPLE MINIMA POINTS FOUND, RETRY WITH SMALLER ALLOWABLE ERROR
C
      KTHR = KTHR - 1
      IF(KTHR.GE.1) GO TO 10
   50 CONTINUE
C
C     MULTIPLE MINIMA POINTS AT SMALLEST ALLOWABLE ERROR
C     TAKE THE SMALLEST DISPLACEMENT TO BE THE ESTIMATE
C
      MMIN = 242
      DO 60 INUM = 1,NCOUNT
      MOT = (IPOS(INUM,1)-11)*(IPOS(INUM,1)-11)+(IPOS(INUM,2)-11)*
     #       (IPOS(INUM,2)-11)
      IF(MOT.GT.MMIN) GO TO 60
C
C     SET THE COUNTERS TO THE SMALLEST DISPLACEMENT
C
      MMIN = MOT
      LL(1) = IPOS(INUM,1)
      LL(2) = IPOS(INUM,2)
   60 CONTINUE
   70 CONTINUE
      IMIN = E(LL(1),LL(2))
      IF(IMIN.LE.ITHR .OR. ICOUNT.EQ.2) RETURN
      ICOUNT = 2
C
C     SET FULL RANGE METRIC POINTERS
C
      ISTART = IRANGE(1)
      ISTOP  = IRANGE(2)
      JSTART = IRANGE(3)
      JSTOP  = IRANGE(4)
      GO TO 10
   80 CONTINUE
      KTEST = N3**2
```

```
C      IMIN = 2000000
C
C*********************************************************
C*                                                       *
C*      SECTION FOR BORDER BLOCKS                        *
C*                                                       *
C*********************************************************
C
       DO 110 I=ISTART,ISTOP
       IM1 = I - 1
       DO 110 J=JSTART,JSTOP
C
C      KCOUNT IS USED FOR THE THRESHOLD EXCEEDING COUNTING METHOD
C      KSUM IS USED FOR SUM OF SQUARES OF DIFFERENCE METHOD
C      KK IS A COUNTER FOR NUMBER OF PIXEL IN BORDER BLOCKS
C
       KCOUNT = 0
       KSUM = 0
       KK = 0
       JM1 = J - 1
       DO 90 K=1,N3
       IK = K + IM1
C
C      DETERMINE IF OUTSIDE OF BORDER
C
       IF(IK.LT.KCON(2) .OR. IK.GT.KCON(4)) GO TO 90
       DO 90 L=1,N3
       JL = L + JM1
C
C      DETERMINE IF OUTSIDE OF BORDER
C
       IF(JL.LT.KCON(1) .OR. JL.GT.KCON(3)) GO TO 90
       KK = KK + 1
       KDIF = IABS(A(L,K) - B(JL,IK))
       KSUM = KSUM + KDIF
       IF(KDIF.GT.KTHR)KCOUNT = KCOUNT + 1
   90  CONTINUE
       XKK = KK
       IF(KK.NE.0) GO TO 100
C
C      NO PIXEL OVERLAP
C
       KSUM = 255 * IISIZE
       KCOUNT = IISIZE
       KK = IISIZE
  100  CONTINUE
C
C      AT LEAST SOME PIXEL OVERLAP
C
       IF(ICOUNT.EQ.1) GO TO 105
       IF(KK.EQ.IISIZE) GO TO 105
C
C      ADJUST FOR NON-FULL BLOCK
C
       KSUM = IFIX((SIZE/FLOAT(KK))*KSUM)
       COUNT = KCOUNT
       KCOUNT = INT(COUNT*(SIZE/XKK) + .5)
  105  D(J,I) = KCOUNT
       E(J,I) = KSUM
       IF(KCOUNT.GE.KTEST) GO TO 110
C
C      RESET COUNTING METHOD POINTERS AND MINIMUM
C
       KTEST = KCOUNT
       LL(1) = J
       LL(2) = I
 C     IMIN = KSUM
  110  CONTINUE
C
C      TEST FOR POSSIBLE MULTIPLE MINIMA (NCOUNT = #MINIMUMS)
C
       NCOUNT = 0
       DO 120 I=ISTART,ISTOP
       DO 120 J=JSTART,JSTOP
       IF(D(I,J) .NE. KTEST) GO TO 120
```

```
      NCOUNT = NCOUNT + 1
      IPOS(NCOUNT,1) = I
      IPOS(NCOUNT,2) = J
  120 CONTINUE
      IF(NCOUNT.EQ.1) GO TO 140
C
C      MULTIPLE MINIMA POINTS FOUND, RETRY WITH SMALLER ALLOWABLE ERROR
C
      KTHR = KTHR - 1
      IF(KTHR .GE. 1) GO TO 80
C
C     MULTIPLE MINIMA POINTS AT SMALLEST ALLOWABLE ERROR
C     TAKE SMALLEST DISPLACEMENT TO BE DISPLACEMENT
C
      MMIN = 242
      DO 130 INUM = 1,NCOUNT
      MOT = (IPOS(INUM,1) - 11)*(IPOS(INUM,1) - 11)
     #      (IPOS(INUM,2) - 11)*(IPOS(INUM,2) - 11)
      IF(MOT.GT.MMIN) GO TO 130
C
C      SET COUNT TO SMALLEST DISPLACEMENT
C
      MMIN = MOT
      LL(1) = IPOS(INUM,1)
      LL(2) = IPOS(INUM,2)
  130 CONTINUE
  140 CONTINUE
      IMIN = E(LL(1),LL(2))
      IF(IMIN.LE.ITHR .OR. ICOUNT.EQ.2) RETURN
      ICOUNT = 2
C
C     SET FULL RANGE METRIC POINTERS
C
      ISTART = IRANGE(1)
      ISTOP  = IRANGE(2)
      JSTART = IRANGE(3)
      JSTOP  = IRANGE(4)
      GO TO 80
      END
```

SUBROUTINE MMUL

```
      SUBROUTINE MMUL(A,B,C,NA,MA,MB,N1,N2,N3)
C*************************************************************
C*                                                           *
C*      SUBROUTINE:MMUL                                      *
C*      PURPOSE:                                             *
C*              THIS SUBROUTINE IS USED TO MULTIPLY TWO      *
C*              ARRAYS TO FORM A THIRD IN THE FORMµ          *
C*                 (A)*(B) = (C)                             *
C*                                                           *
C*      EXPLANATION OF CALL VARIABLES:                       *
C*              A       - I - 1ST REAL INPUT MATRIX          *
C*              B       - I - 2ND REAL INPUT MATRIX          *
C*              C       - O - REAL OUTPUT MATRIX             *
C*              NA      - I - 1ST SIZE VARIABLE OF A AND C   *
C*              MA      - I - 2ND SIZE VARIABLE OF A, 1ST B  *
C*              MB      - I - 2ND SIZE VARIABLE OF B AND C   *
C*              N1      - I - USABLE FORM OF NA              *
C*              N2      - I - USABLE FORM OF MA              *
C*              N3      - I - USABLE FORM OF MB              *
C*                                                           *
C*      SUBROUTINES NEEDED:                                  *
C*              NONE                                         *
C*                                                           *
C*************************************************************
      REAL A(NA,MA),B(MA,MB),C(NA,MB)
      DO 20 I=1,N1
      DO 20 J=1,N3
      T = 0.
      DO 10 K=1,N2
   10 T = T + A(I,K)*B(K,J)
   20 C(I,J) = T
      RETURN
      END
```

SUBROUTINE MPLOT

```
      SUBROUTINE MPLOT(F,XE,ISIZE,IPLT,IMAGE,IBLK,JBLK,LOC,NGRD)
C***********************************************************
C*                                                         *
C*      SUBROUTINE MPLOT                                   *
C*                                                         *
C*      PURPOSE:                                           *
C*              THIS SUBROUTINE IS USED TO PLOT A 3-D DATA-*
C*              BASE IN TWO DIMENSIONS.  IN THIS CASE IT IS*
C*              USED TO PLOT DIFFERENCE METRIC (ACTUALLY AN*
C*              INVERTED SCALED VERSION).                  *
C*                                                         *
C*      EXPLANATION OF CALL VARIABLES:                     *
C*              F      - I - METRIC ARRAY TO BE PLOTTED    *
C*              XE     - T - REAL WORK ARRAY VERSION OF F  *
C*              ISIZE  - I - SIZE OF SQUARE METRIC         *
C*              IPLT   - I - DISPLAY DEVICE FLAG           *
C*                     = 0   PLOT TO 4662 PLOTTER          *
C*                     = 1   PLOT ON 4025 DISPLAY          *
C*              IMAGE  - I - INTEGER VALUE OF FRAME NUMBER *
C*              IBLK   - I - BLOCK LOCATION VARIABLE       *
C*              JBLK   - I - BLOCK LOCATION VARIABLE       *
C*              LOC    - I - MINIMUM VALUE LOCATION        *
C*              NGRD   - I - ARRAY REQUIRED BY THE 3D PLOT *
C*                                                         *
C*      SUBROUTINES NEEDED:                                *
C*              INVERT - CHANGES ORDER AND SCALE 0-1       *
C*              ERASE  - CLEARS 4025 SCREEN                *
C*              FACTOR - SCALES DATA BEFORE PLOTTING       *
C*              PUR    - 3-D PLOT ROUTINE DRIVER           *
C*              PLOTS  - CALCOMP PLOTTING SOFTWARE PACKAGE *
C*                                                         *
C***********************************************************
      INTEGER F(ISIZE,ISIZE),LOC(4)
      REAL XE(ISIZE,ISIZE),NGRD(4),XY(2,6)
      LUNIT = 6
C
      CALL INVERT(F,XE,ISIZE)
C
C     SET 3D PLOT PARAMETERS - LOOK AT SUBROUTINE PUR FOR MORE DETAILS
C
      NGRD(1) = 0
      NGRD(2) = 1
      NGRD(3) = 1
      NGRD(4) = 1
      X = 0.0
      Y = 5.2
      IF(IPLT.EQ.1) CALL ERASE
      IF(IPLT.EQ.0)CALL PLOT(X,Y,-3)
      IF(IPLT.EQ.0)CALL PLOTS(IBUF,1,15)
      IF(IPLT.EQ.1)CALL FACTOR(.400)
      IF(IPLT.EQ.0)CALL FACTOR(.40)
      CALL PUR(XE,ISIZE,ISIZE,ISIZE,XY,0,NGRD,15)
      IF(IPLT.EQ.1)CALL FACTOR(.40)
      IF(IPLT.EQ.0)CALL FACTOR(.40)
      CALL PLOT(0.0,4.3,-3)
C     CALL SYMBOL(1.,6.9,.20,'FRAME',0.0,5)
C     CALL SYMBOL(1.,6.6,.20,'IBLOCK',0.,6)
C     CALL SYMBOL(1.,6.3,.20,'JBLOCK',0.,6)
      XFRAME = IMAGE + 1
      XBLOCK = JBLK
      YBLOCK = IBLK
C     CALL NUMBER(2.6,6.9,.20,XFRAME,0.0,-1)
C     CALL NUMBER(2.6,6.6,.20,XBLOCK,0.0,-1)
C     CALL NUMBER(2.6,6.3,.20,YBLOCK,0.0,-1)
C     CALL SYMBOL(1.,6.,.20,'X SHIFT',0.,7)
C     CALL SYMBOL(1.,5.70,.20,'Y SHIFT',0.,7)
      XSHIFT = LOC(1) - 11
      YSHIFT = LOC(2) - 11
C     CALL NUMBER(2.6,6.0,.20,XSHIFT,0.0,-1)
C     CALL NUMBER(2.6,5.7,.20,YSHIFT,0.0,-1)
C     IF(IPLT.EQ.1) CALL SYMBOL(0.0,0.0,.10,'.',0.0,1)
      IF(IPLT.EQ.1) CALL ANMODE
      IF(IPLT.EQ.1) CALL TSEND
      IF(IPLT.EQ.0) CALL PLOT(10.,10.,999)
      WRITE(LUNIT,400)
C
C     READ DUMMY ARGUMENT FOR PLOTTER DELAY
C
      READ(5,410)INSWER
  400 FORMAT(1X,'/*INPUT ANY SINGLE DIGIT NUMBER TO CONTINUE')
  410 FORMAT(I1)
      RETURN
      END
```

SUBROUTINE PRED

```
      SUBROUTINE PRED(C,D,ES,ICD,IS,IXF,IYF,STD,AVG,QOFF,MIN,COEF,IFLAG,
     # NBITS,RETCOD,F,DIST,LVLARA)
C***************************************************************
C*                                                             *
C*      SUBROUTINE:PRED                                        *
C*      PURPOSE:                                               *
C*              THIS SUBROUTINE IS USED TO CALCULATE THE       *
C*              CURRENT IMAGE FRAME FROM INFORMATION           *
C*              CONTAINED IN THE PREVIOUS IMAGE FRAME AND      *
C*              THE PREDICTION COEFICIENTS GENERATED FOR       *
C*              THE CURRENT FRAME                              *
C*                                                             *
C*      EXPLANATION OF CALL VARIABLES:                         *
C*              C      - I - INTEGER ARRAY FOR CURRENT         *
C*                           IMAGE BLOCK                       *
C*              D      - I - INTEGER ARRAY FOR PREVIOUS        *
C*                           ESTIMATED IMAGE                   *
C*              ES     - 0 - INTEGER ARRAY FOR CURRENT         *
C*                           ESTIMATED IMAGE                   *
C*              ICD    - I - SIZE OF THE D MATRIX              *
C*              IS     - I - SIZE OF THE ES AND C MATRICES     *
C*              IXF    - I - X OFFSET VARIABLE                 *
C*              IYF    - I - Y OFFSET VARIABLE                 *
C*              STD    - 0 - STANDARD DEV OF THE ERROR         *
C*              AVG    - 0 - AVERAGE OF THE ERROR              *
C*              QOFF   - I - ARRAY OF POINTER VALUES           *
C*              MIN    - I - MINIMUM QUANDRANT NUMBER          *
C*              COEF   - I - PREDICTION COEFFICIENT VECTOR     *
C*              IFLAG  - I - INTEGER DISPLACEMENT FLAG         *
C*              NBITS  - 0 - DATA RATE COUNTER                 *
C*              RETCOD - 0 - LARGE ERROR/BLKSIZE BIT           *
C*              F      - 0 - OUTPUT WORK ARRAY (ERROR?)        *
C*              DIST   - I - PREDICTOR GAIN (IN DB)            *
C*                                                             *
C*      SUBROUTINES NEEDED:                                    *
C*              QUANTZ - SETS VALUES FOR ADAPTIVE QUANTIZER*
C*                                                             *
C***************************************************************
      INTEGER C(IS,IS),D(ICD,ICD),QOFF(4,4,2),ES(IS,IS),RETCOD
      INTEGER F(IS,IS),G(16,16),LVLARA(1)
      REAL XLEVEL(128),XOUT(128),COEF(4),DIST(1)
      COMMON/THRESH/ STDERR,AVGERR,SNRSET
      SIZE = IS**2
      RETCOD = 0
      KSTART = 1
      KSTOP = 4
C
C     CHECK FOR INTEGER DISPLACEMENT
C
      IF(IFLAG.NE.0) GO TO 10
      KSTART = 3
      KSTOP = 3
   10 KA = 0
      NA = 0
      KV = 0
      NV = 0
      DO 30 J=1,IS
      JJ = J + IXF
      DO 30 I=1,IS
      II = I + IYF
      TEMP = 0.0
C
C     MAKE PREDICTION BASED ON PREDICTION COEFFICIENTS
C
      DO 20 M=KSTART,KSTOP
      TEMP = TEMP + COEF(M)*FLOAT(D(II+QOFF(MIN,M,1),JJ+QOFF(MIN,M,2)))
   20 CONTINUE
      ES(I,J) = TEMP + .5
      IERROR = C(I,J) - ES(I,J)
      F(I,J) = IERROR
      KA = KA + IERROR
      KV = KV + IERROR*IERROR
      NV = NV + C(I,J)*C(I,J)
      NA = NA + C(I,J)
   30 CONTINUE
      AVGK = KA
      AVGN = NA
      VARK = KV
      VARN = NV
      VARKK = (VARK - (AVGK*AVGK/SIZE))/(SIZE-1.)
      VARNN = (VARN - (AVGN*AVGN/SIZE))/(SIZE-1.)
      IF(VARNN.EQ.0.0)SNR = SNRSET
      IF(VARNN.EQ.0.0)GO TO 15
      SNR = 10.*ALOG10(VARNN/VARKK)
   15 STD = SQRT(VARKK)
      AVG = AVGK/SIZE
      IF(STD.LT. STDERR .AND. ABS(AVG).LT.AVGERR) GO TO 70
C
C     RETURN AND RETRY IF:
C      1) LARGE ERROR FOR 16X16 BLOCK
C      2) LARGE ERROR FOR INTEGER DISPLACEMENT ON 8X8 BLOCK
C
      IF(IS.EQ.16) RETURN
      IF(IFLAG.EQ.0)RETCOD = 1
      IF(IFLAG.EQ.0)RETURN
C
C     CORRECT FOR LARGE ERROR WITH BLOCK ADAPTIVE QUANTIZER
C     DETERMINE QUANTIZER GAIN REQUIRED TO OBTAIN SET SNR
C
      REQGAN = SNRSET - SNR
C
C     QUANTIZE THE MEAN AND VARIANCE BEFORE ERROR QUANTIZATION
C
      AVG1 = (FLOAT(INT(AVG*2.0 + SIGN(.5,AVG))))/2.0
      STD1 = (FLOAT(INT(STD*8.0 + .5)))/8.0
C
C     CHECK FOR QUANTIZER LEVEL OVERFLOW
C
C     IF(ABS(AVG1).GT.32.0)WRITE(6,55)
C     IF(STD1.GT.64.)WRITE(6,56)
   55 FORMAT(1X,'AVERAGE ERROR TO LARGE, LOOK AT PRED.')
   56 FORMAT(1X,'STANDARD DEVIATION ERROR TO LARGE, LOOK AT PRED.')
C     IF(ABS(AVG1).GT.32.0)AVG1=32.*SIGN(1.,AVG)
C     IF(STD1.GT.64.0)STD1=64.0
C
C     SUBROUTINE QUANTZ WILL DETERMINE THE NUMBER OF LEVELS REQUIRED
C
C     REQGAN = 10.*ALOG10(VARKK/3.)
      CALL QUANTZ(REQGAN,AVG1,STD1,LEVEL,XLEVEL,XOUT,DIST)
      NBITS = NBITS + 23 + INT(SIZE*ALOG(FLOAT(LEVEL))/ALOG(2.) + 1.0)
      LVLARA(LEVEL) = LVLARA(LEVEL) + 1
C    IF(LEVEL.LE.128)WRITE(6,11)LEVEL,REQGAN,AVG1,STD1,VARNN,VARKK,AVGK
   11 FORMAT(1X,'REQUIRES',I3,' BITS',6F10.4)
C
C     7 BITS FOR THE MEAN
C     9 BITS FOR THE STD
C     7 BITS FOR THE NUMBER OF LEVELS
C
      DO 60 J=1,IS
      DO 60 I=1,IS
      IER = C(I,J) - ES(I,J)
      X = IER
      DO 40 K=1,LEVEL
      IF(X.GT.XLEVEL(K)) GO TO 40
      IQUAN = INT(XOUT(K) + SIGN(.5,XOUT(K)))
      GO TO 50
   40 CONTINUE
   50 CONTINUE
      ES(I,J) = ES(I,J) + IQUAN
      G(I,J) = C(I,J) - ES(I,J)
   60 CONTINUE
   70 CONTINUE
C
C     RESTRICT OUTPUT FOR 0-255.
C
      DO 80 J=1,IS
      DO 80 I=1,IS
      IF(ES(I,J).GT.255) ES(I,J) = 255
```

```
    IF(ES(I,J).LT.0) ES(I,J) = 0
80  CONTINUE
    RETURN
    END
```

SUBROUTINE TRANSP

```
      SUBROUTINE TRANSP(A,B,N1,N2,N3,N4)
C*************************************************************
C*                                                           *
C*      SUBROUTINE:TRANSP                                    *
C*      PURPOSE:                                             *
C*              THIS SUBROUTINE IS USED TO TRANSPOSE ARRAYS*
C*                                                           *
C*      EXPLANATION OF CALL VARIABLES:                       *
C*              A       - I - INPUT DATA ARRAY               *
C*              B       - O - OUTPUT REAL DATA ARRAY         *
C*              N1      - I - 1ST ACTUAL SIZE VARIABLE OF A *
C*                            IN CALLING PROGRAM             *
C*              N2      - I - 2ND ACTUAL SIZE VARIABLE OF A *
C*                            IN CALLING PROGRAM             *
C*              N3      - I - 1ST USING SIZE VARIABLE        *
C*              N4      - I - 2ND USING SIZE VARIABLE        *
C*                                                           *
C*      SUBROUTINES NEEDED:                                  *
C*              NONE                                         *
C*                                                           *
C*************************************************************
      REAL A(N1,N2),B(N2,N1)
      DO 10 I=1,N3
      DO 10 J=1,N4
   10 B(J,I) = A(I,J)
      RETURN
      END
```

SUBROUTINE QUANTI

```
      SUBROUTINE QUANTI(DIST)
C***********************************************************
C*                                                         *
C*     SUBROUTINE QUANTI                                   *
C*     PURPOSE:                                            *
C*             THIS SUBROUTINE IS USED TO GENERATE THE     *
C*             FUNCTIONAL VALUES FOR A NORMAL(0,1)         *
C*             DISTRIBUTION.                               *
C*                                                         *
C*     EXPLANATION OF CALL VARIABLES:                      *
C*             DIST   - O - ARRAY CONTAINING GENERATED     *
C*                          DATA                           *
C*                                                         *
C*     SUBROUTINES NEEDED:                                 *
C*             MDNRIS - GENERATE VALUES FOR GUASSIAN DIST  *
C*                                                         *
C***********************************************************
      REAL X(1001),F(1001),OUT(1003),XOUT(128),XLEVEL(128),DIST(1)
C
C     GENERATE THE VALUES FOR THE N(0,1) DISTRIBUTION
C
      DO 10 I=1,1001
      X(I) = (FLOAT(I) - 501.)/100.
      F(I) = .398942*EXP(-.5*(X(I)*X(I)))
   10 CONTINUE
      DIST(1) = 0.
      DO 90 LEVEL=2,128
      XLEVL = LEVEL
      START = 1./XLEVL
      NLEVL1 = LEVEL - 1
      PERCNT = START
      DO 20 I=1,NLEVL1
      CALL MDNRIS(PERCNT,XLEVEL(I),IER)
   20 PERCNT = PERCNT + START
      XLEVEL(LEVEL) = 1000.
      BEGIN = START/2.
      PERCNT = BEGIN
      DO 30 I=1,LEVEL
      CALL MDNRIS(PERCNT,XOUT(I),IER)
   30 PERCNT = PERCNT + START
      DO 40 I=1,1001
   40 OUT(I) = 0.0
      DO 70 I=1,1001
      VALUE = (FLOAT(I-501))/100.
C-----DETERMINE WHICH LEVEL IT FALLS WITHIN
      DO 50 II=1,LEVEL
      IF(VALUE.GT.XLEVEL(II)) GO TO 50
      K = II
      GO TO 60
   50 CONTINUE
   60 CONTINUE
C-----XOUT(K) IS THE AMOUNT OF SHIFT
      IVAL = INT((VALUE - XOUT(K))*100 + 501.5)
      OUT(IVAL) = OUT(IVAL) + F(I)
   70 CONTINUE
C-----CALCULATE THE DISTORTION
      DIS = 0.0
      DO 80 I=1,1000
      XX = X(I)*X(I)
   80 DIS = DIS + XX*(OUT(I) + OUT(I+1))*.005
   90 DIST(LEVEL) = 10.*ALOG10(1./DIS)
      RETURN
      END
```

SUBROUTINE QUANTZ

```
      SUBROUTINE QUANTZ(REQGAN,XBAR,STD,LEVEL,XLEVEL,XOUT,DIST)
C************************************************************
C*                                                          *
C*     SUBROUTINE:QUANTZ                                    *
C*     PURPOSE:                                             *
C*             THIS SUBROUTINE IS USED TO SET UP THE        *
C*             QUANTIZER LEVELS AND THRESHOLDS              *
C*                                                          *
C*     EXPLANATION OF CALL VARIABLES:                       *
C*             REQGAN- I - REQUIRED GAIN OF THE QUANTIZER *
C*             XBAR  - I - MEAN OF GUASSIAN DISTRIBUTION   *
C*             STD   - I - VARIANCE OF GAUSIAN DIST.        *
C*             LEVEL - O - NUMBER OF LEVELS USED FOR CODE *
C*             XLEVEL- O - DISTRIBUTION THRESHOLDS          *
C*             XOUT  - O - QUANTIZED OUTPUT LEVELS          *
C*             DIST  - I - ARRAY CONTAINING GAINS FOR EACH*
C*                         NUMBER OF LEVELS                 *
C*                                                          *
C*     SUBROUTINES NEEDED:                                  *
C*             MDNRIS - ROUTINE TO FIND AREA UNDER CURVE   *
C*                      FOR A NORMAL(0,1) GAUSSIAN DIST.    *
C*                      (IMSL LIBRARY)                      *
C*                                                          *
C************************************************************
      REAL XOUT(128),XLEVEL(128),DIST(1)
C
C     DETERMINE THE NUMBER OF LEVELS REQUIRED FOR ERROR TRANSMISSION
C     BASED ON THE VALUE FOR REQGAN
C
      II = 1
      DO 5 I=2,128
      II = II + 1
      IF(REQGAN.LT.DIST(I)) GO TO 6
    5 CONTINUE
    6 CONTINUE
C
C     DETERMINE THE THRESHOLD LEVELS
C
      LEVEL = II
      XLEVL = LEVEL
      START = 1./XLEVL
      NLEVL1 = LEVEL - 1
      PERCNT = START
      DO 10 I=1,NLEVL1
      CALL MDNRIS(PERCNT,XLEVEL(I),IER)
      XLEVEL(I) = XLEVEL(I)*STD + XBAR
      PERCNT = PERCNT + START
   10 CONTINUE
C
C     DETERMINE THE OUTPUT LEVELS
C
      XLEVEL(LEVEL) = 100000.
      BEGIN = START/2.
      PERCNT = BEGIN
      DO 20 I=1,LEVEL
      CALL MDNRIS(PERCNT,XOUT(I),IER)
      XOUT(I) = XOUT(I)*STD + XBAR
      PERCNT = PERCNT + START
   20 CONTINUE
      RETURN
      END
```